UiA Universitetet i Agder

Master

Programmering i matematikk

Hvordan kan programmering bli en integrert del av matematikkundervisningen på ungdomstrinnet?

THOMAS CHRISTOPHER CANDASAMY

VEILEDER

Linda G. Opheim

Universitetet i Agder, 2021

Fakultet for teknologi og realfag

Institutt for matematikk

Forord

Denne masteroppgaven markerer siste ledd i fullførelsen av min mastergrad i matematikkdidaktikk ved Universitetet i Agder. De siste tre årene har jeg via regjeringens satsing *Kompetanse for kvalitet* hatt muligheten til å videreutdanne meg i et fag jeg er svært engasjert i. Disse tre årene har jeg studert på deltid ved siden av mitt daglige virke som ungdomsskolelærer; det har vært tøft og tidkrevende men jeg har fått utvikle meg selv som lærer og det har vært givende. Jeg håper arbeidet mitt kan være til hjelp for andre og bidra med inspirasjon i arbeidet med å integrere programmering som en del av matematikkfaget i skolen.

Det er flere jeg ønsker å takke i forbindelse med dette arbeidet. Aller først vil jeg rette en stor takk til mine ledere og kollegaer på min arbeidsplass. De har støttet meg, heiet på meg og lagt til rette for at jeg kunne gjennomføre masterstudiet. Deres støtte, forståelse og tillit har gitt meg motivasjon og styrke i krevende tider.

Jeg ønsker å takke rådgiver Trine Engeland ved institutt for matematiske fag på UiA. Hun har hjulpet med opptak og tilrettelegging av studiet for en deltidsstudent. Hver gang jeg har lurt på noe studierelatert har hun besvart mailene mine både raskt, utfyllende og med en vennlig tone.

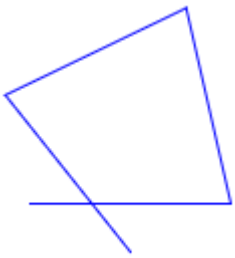
En spesiell takk ønsker jeg å rette til veilederen min Linda Gurvin Opheim. Hun har vist forståelse for min arbeids- og livssituasjon og støttet mine prioriteringer. Hennes raske tilbakemeldinger og konstruktive kritikk har vært helt avgjørende for å komme i mål med denne oppgaven.

Klassene jeg har besøkt og lærerne og elevene som har meldt seg frivillige til å være en del av forskningen min setter jeg utrolig stor pris på. Uten deres bidrag ville det ikke blitt noe av denne oppgaven. Elevene som har deltatt i intervjuer har velvillig delt av seg selv og sine tanker og dette har jeg lært mye av.

En siste takk går til den fantastiske familien min. Selv om jeg føler jeg har vært flink til å balansere arbeids- og familielivet, vet jeg at dere ikke alltid har følt det på samme måte. Takk til min kone Elisabeth som har gitt meg tillatelse til å studere, og som har tatt meg med ut på tur og sørget for at jeg har andre ting å tenke på enn bare studier. Takk til mine døtre, Sarah og Nora, som leker med barbiedukker rett utenfor kontordøra mi. Å høre dere leke sammen har gitt meg uendelig mange smil om munnen. Dere har vært den største motivasjonsfaktoren for meg i dette arbeidet og jeg er glad for at jeg nå kan tilbringe enda mer tid sammen med dere.

Kristiansand, april 2021

Thomas Christopher Candasamy



Sammendrag

Temaet for denne masteroppgaven er programmering i matematikkundervisningen. På grunn av økende digitalisering i samfunnet har det lenge vært argumentert for at skolen må gi elevene mer opplæring i programmering og digital teknologi. Høsten 2020 trådte fagfornyelsen i kraft; alle læreplaner for fag i grunnskolen og videregående opplæring ble fornyet. Alle fag har fått kjerneelementer; det viktigste elevene skal lære. I tillegg er det lagt gjennomgående stor vekt på dybdelæring i alle fag. Det var innholdet i fagene som ble nytt, fag- og timefordelingen forble uendret. Noe av det som var nytt i matematikk var programmering.

Meningene om programmering i matematikk er delte. Mange matematikklærere er misfornøyde og mener programmering vil føre til mer stofftrengsel og mindre tid til dybdelæring. Denne oppfattelsen deler ikke jeg og har ønsket å undersøke dette nærmere. Programmering trenger ikke å komme *i tillegg* til de andre fagområdene i matematikk, men kan læres parallelt. Problemstillingen jeg da har valgt er:

Hvordan kan programmering bli en integrert del av matematikkundervisningen på ungdomstrinnet?

I teoridelen av denne oppgaven avklarer jeg noen begreper som blir brukt videre i dette arbeidet. Først forklarer jeg hva som menes med dybdelæring og deretter hva dybdelæring i matematikk kan være. Videre beskriver jeg hvordan jeg definerer programmering som sammensatt av både algoritmisk tenking (prosesser) og algoritmer (produkter) i denne oppgaven.

For å undersøke problemstillingen min har jeg valgt å utvikle tre undervisningsopplegg i matematikk hvor programmering er hovedaktiviteten. Alle tre opplegg er blitt prøvd ut i klasserommet og jeg har gjennomført to oppgavebaserte intervju med elevpar som arbeider med oppgavene. I ettertid har jeg analysert datamaterialet mitt i to omganger. Først analyserer jeg om elevene faktisk har arbeidet med programmering, til dette bruker jeg kjennetegn på algoritmisk tenking (Utdanningsdirektoratet, 2019a). Deretter analyserer jeg datamaterialet på nytt i lys av kjerneelementene i matematikk og undersøker om elevene også har arbeidet med faget slik det beskrives i læreplanen (Utdanningsdirektoratet, 2020b). Dermed har jeg grunnlag for å uttale meg om programmeringskompetanse og kompetanse i matematikk kan utvikles samtidig.

Forskningen min viser at kompetanse i programmering og matematikk kan utvikles parallelt, det behøver ikke å komme *i tillegg*, men kan være en alternativ måte å arbeide med faget på. Ved å arbeide med programmering settes prosessene til den algoritmiske tenkeren i gang. Disse prosessene overlapper i stor grad med beskrivelsene av kjerneelementene i matematikk. Programmering kan derfor være en glimrende måte å arbeide med matematikk på. Elevene jeg har intervjuet har ikke bare lært *om* programmering, de har arbeidet *som* programmerere og fått ny matematisk kunnskap gjennom dette.

Abstract

The theme for this master thesis is programming in mathematics education. Due to an increasing digitalization of our society, there has grown forth a need to include more programming and digital technology in our schools. In the autumn of 2020, all curriculums for subjects in Norwegian primary and secondary schools were updated; renewed. All subjects received descriptions of the core elements pertaining to that subject which described what it was most important for pupils to learn in each subject. The number of subjects and time allocated to each subject remained unchanged. One of the differences in mathematics was the inclusion of programming.

There are split views upon programming in mathematics. Many mathematics teachers are unhappy and believe the inclusion will lead to too much to teach and not enough time to achieve deep learning. I do not share this view and wanted to devote my research to investigating this notion. I do not believe that programming has to be taught *in addition* to all other subject areas. I believe programming and other mathematics topics can be taught simultaneously. The research question I have chosen is:

How can programming become an integrated part of mathematics education in lower secondary school?

In the literature part of this thesis, I clarify some of the concepts used in the rest of this paper. First, I explain what is meant with deep learning and then what deep learning in mathematics can be. I go on to describe how I define programming as composed of Computational Thinking (processes) and algorithms (products) for the use in this paper.

In order to investigate my research question, I have chosen to develop three lesson plans in mathematics where programming is the main activity. All three lesson plans have been tested in the classroom and I have also conducted two task-based interviews with pairs of pupils working on the tasks I have developed. Afterwards, I have analysed my collection of data twice. The first time, I review whether the pupils have actually been working with programming. To do this, I relate their work to descriptions of Computational Thinking. The second time, I review the data once more but this time look for evidence that the pupils have been working with mathematics the way it has been described in the subject's core elements. By looking at both the programming and the mathematics involved during the lessons, I can comment on whether competencies in both areas are developing simultaneously.

My research shows exactly that, competencies in programming and mathematics can be developed simultaneously. Programming does not have to come *in addition*, but can provide an alternative way of working with the subject matter. By working with programming tasks, the processes related to the computational thinker are activated. These processes greatly coincide with the core elements in mathematics. Therefore, programming can be an excellent way of working with mathematics. The pupils I have interviewed have not only learned *about* programming, they have worked *as* programmers and gained new mathematical insights during the process.

Innhold

1 Innledning

Denne mastergradsoppgaven handler om programmering i matematikk. Jeg skriver denne oppgaven som et ledd i arbeidet med å fullføre en mastergrad i matematikkdidaktikk ved Universitetet i Agder. Studiet er 2-årig, men jeg gjennomfører det på deltid over tre år. Samtidig arbeider jeg på en ungdomsskole hvor jeg blant annet underviser i matematikk og programmering som valgfag.

Da jeg startet arbeidet med denne oppgaven i 2018 hadde det blitt bestemt at programmering skulle bli en del av kompetansen norske skoleelever tilegner seg i løpet av skolegangen. Når samfunnet endrer seg, må skolen følge etter. Store deler av arbeidslivet er preget av den teknologiske utviklingen (Kunnskapsdepartementet, 2016, s. 6) og dermed må elevene få utvikle tilstrekkelig digital kompetanse mens de går på skolen hvis de skal være rustet for dette. Det holder ikke å være passive konsumenter av teknologien rundt, elevene må i større grad bli aktive skapere. «Det må legges til rette for at elevene kan få lære og forstå de grunnleggende prinsippene som ligger bak teknologien, slik at de kan ta kontroll over den og kreativt utvikle den til sitt eget og fellesskapets beste» (Sanne et al., 2016, s. 7). I fremtiden vil mange av dagens elever være involvert i utviklingen av teknologi som er viktig for samfunnet og det har vært en økende trend i Europa å få integrert programmeringskompetanse i skolenes læreplaner (Balanskat & Engelhardt, 2015). En ekspertgruppe utnevnt av Utdanningsdirektoratet i 2016 anbefalte å opprette et nytt obligatorisk teknologifag i grunnskolen som skulle gi elevene opplæring i bl.a. programmering og digital teknologi (Sanne et al., 2016).

Fagfornyelsen trådte i kraft fra høsten 2020, hvor alle læreplaner for fag i grunnskolen og videregående opplæring ble fornyet (Utdanningsdirektoratet, 2020a). Det var innholdet i fagene som ble nytt, fag- og timefordelingen forble uendret. Dermed måtte programmering inn i eksisterende fag, og ikke bli et eget fag slik mange lærere ytret et ønske om (Utdanningsdirektoratet, 2017b). Det ble bestemt at programmering skulle inn i matematikk, naturfag, kunst og håndverk og musikk, men hvor hovedvekten ble lagt til matematikkfaget. Kompetanser som får økende betydning i et stadig mer digitalisert yrkesliv er blant annet problemløsing, kreativitet og logisk tenking. Dette er sentrale kompetanser i både matematikk og programmering. Programmerere må kunne analysere, forstå og løse problemer ved å utvikle og verifisere algoritmer. Disse prosessene er sterkt tilknyttet matematikk og flere land har derfor valgt å integrere programmering i læreplanen for matematikk (Forsström & Kaufmann, 2018, s. 19). Dette er ikke uproblematisk og flere utfordringer står i kø. Først og fremst utfordres matematikklærerne, for de har kanskje ikke noe tidligere kjennskap til programmering. Da forslag til kjerneelementene i matematikk ble sendt ut på høring var det pålegget om programmering i matematikk som fikk mest oppmerksomhet. De fleste innspillene gikk imot dette; det ble argumentert for at programmering ville føre til større stofftrengsel og mindre tid til å gå i dybden (Utdanningsdirektoratet, 2017c). Mange lærere var kritiske; faget var allerede overfylt og nå skulle de måtte undervise programmering *i tillegg*.

Fagfornyelsen har ikke bare endret på innholdet i fagene, men også på læreplanstrukturen og overordnet del av læreplanverket. Det står at elevene skal utvikle dybdelæring, i motsetning til overflatelæring. Dette må det forskes på; hvordan kan dybdelæring realiseres i klasserommet? I tillegg har alle fag fått kjerneelementer; det viktigste elevene skal lære i hvert fag. «Kjerneelementene skal prege innholdet og

progresjonen i læreplanene og bidra til at elevene over tid utvikler forståelse av innhold og sammenhenger i faget» (Utdanningsdirektoratet, 2017a). Kjerneelementene i matematikk er

- Utforsking og problemløsing
- Modellering og anvendelser
- Resonnering og argumentasjon
- Representasjon og kommunikasjon
- Abstraksjon og generalisering
- Matematiske kunnskapsområder

Skoleåret 2016/2017 igangsatte utdanningsdirektoratet et forsøksprosjekt med programmering som valgfag på ungdomstrinnet. Året før hadde jeg tilfeldigvis fullført et nettbasert årsstudium i informatikk, hvor flere av emnene var direkte knyttet til programmering, og med bakgrunn i dette søkte min skole om å bli med i prosjektet. Siden har jeg undervist programmering som valgfag. For meg hører programmering naturlig inn som en gren av matematikken. I arbeidet med programmering, må elevene blant annet opprette og bruke variabler for å løse problemer. I Norge har spesielt algebra vært problematisk for elevene (Utdanningsdirektoratet, 2019c). Gjennom programmering kan en få et annet forhold til variabler og algebraiske uttrykk. Elevene får muligheten til å se hvor nyttig og effektivt det kan være å benytte seg av slike løsninger.

Mange vil nok tenke at programmering er å skrive kode som en datamaskin kan utføre for å få noe til å skje. Og det er nok det som er *produktet* av programmering, men en må ikke glemme *prosessene* som programmering setter i gang.

> «Programmering […] omfatter mer enn å bare skrive programkode som kan kjøres på en datamaskin, det inkluderer også prosessen med å komme fram til denne koden. Det vil si prosessen fra å identifisere et problem og tenke ut mulige løsninger på problemet, til å skrive kode som kan forstås av en datamaskin, og å feilsøke og kontinuerlig forbedre denne koden» (Sevik, 2016, s. 9).

På engelsk skilles det mellom *Computational Thinking (CT)* og *Programming*. «Thinking computationally is not programming […] Simply put, programming tells a computer what to do and how to do it. Computational thinking enables you to work out exactly what to tell the computer to do» (BBC, 2021). På norsk har CT blitt oversatt til *algoritmisk tenking*, men i dagligtale brukes fortsatt programmering om både prosess og produkt. Vi bruker algoritmisk tenking til å analysere, forstå og finne løsninger på problemer slik at vi deretter kan programmere f.eks. en datamaskin til å utføre løsningene. Algoritmisk tenking involverer logikk, algoritmer, dekomposisjon, mønstre, abstraksjon og evalueringer (Utdanningsdirektoratet, 2019a). Her er det stor overlapp med kjerneelementene i matematikk, dette skriver jeg mer om i kapittel 2.3.1 Programmering og matematikk.

Vi vet at programmeringskompetanse er viktig for fremtiden. Vi vet at programmering er sterkt knyttet til matematikk og vil inkluderes i læreplanene i flere land. Det vi ikke vet nok om, er *hvordan* programmering skal integreres i matematikkundervisningen. Hvordan underviser man programmering i matematikk? Jeg håper at inntredenen av programmering i matematikkfaget ikke nødvendigvis trenger å bety *mer* stoff i et

allerede overfylt fag, men en arbeidsmåte som kan svare på noen av kravene som den nye læreplanen etterspør. Jeg formulerer derfor følgende problemstilling:

Hvordan kan programmering bli en integrert del av matematikkundervisningen på ungdomstrinnet?

Jeg ønsker altså å undersøke hvordan programmering kan bli en naturlig del av matematikkundervisningen, og ikke bare noe som kommer i tillegg. Med programmering mener jeg her både prosessene til den algoritmiske tenkeren og programkoden som blir produktet av prosessene.

Planen min er å utvikle tre undervisningsopplegg hvor hovedaktiviteten er knyttet til programmering ved hjelp av ulike programmeringsspråk. Elever fra ungdomstrinnet skal arbeide med oppleggene mens jeg observerer og intervjuer. Målet er å undersøke om elevene, samtidig som de engasjerer seg i programmeringsaktiviteter, også arbeider med kjerneelementene i matematikk. Kan undervisningen tilrettelegges på en slik måte at programmeringskompetanse og kompetanse i matematikk utvikles parallelt? I så fall kan dette få betydning for fremtidens matematikkundervisning.

I kapittel 2 presenterer jeg teorien som ligger til grunn for forskningen min og avklarer noen begreper som blir brukt videre i denne oppgaven. Jeg forklarer hva som menes med dybdelæring og hvordan jeg definerer programmering som sammensatt av både algoritmisk tenking og algoritmer. Deretter viser jeg til tidligere forsøk med programmeringsspråket LOGO i matematikk og viser til hvilke programmeringsspråk jeg kommer til å bruke i mine undervisningsopplegg. Jeg avslutter kapitlet med å beskrive fire fordeler jeg ønsker å tenke med når jeg skal planlegge mine undervisningsøkter.

I kapittel 3 beskriver jeg mitt forskningsdesign og metodene jeg har valgt å bruke. Jeg beskriver prinsipper for oppgavebaserte intervju og videre hvordan jeg foretok utvalget av elever til forskningen. Deretter presenterer jeg hvordan jeg har tenkt å analysere datainnsamlingen min. Kapitlet avsluttes med et blikk på etiske utfordringer og en vurdering av troverdigheten til denne oppgaven.

I kapittel 4 legger jeg fram resultatene og analysene mine. Her presenterer jeg først undervisningsoppleggene jeg har utviklet og hvordan introduksjonen til disse undervisningstimene vil gjennomføres. I neste del analyserer jeg elevarbeidet i forhold til kjennetegn på algoritmiske tenkere. Hensikten er å slå fast at det er programmering elevene arbeider med. Elevarbeidet analyseres så på nytt men nå med fokus på kjerneelementene i matematikk. Hvis elevene både arbeider med programmering og kjerneelementene i matematikk samtidig kan jeg ha belegg for å påstå at begge kompetansene kan utvikles parallelt. Deretter vurderer jeg om undervisningsoppleggene mine har fungert slik jeg hadde tenkt. Jeg avslutter kapitlet med å skrive om holdningene til elevene som var et fenomen jeg i utgangspunktet ikke hadde tenkt å skrive om, men som dukket opp under intervjuene.

I kapittel 5 diskuterer jeg funnene fra mine resultater og analyser. Først skriver jeg om undervisningsoppleggene, deretter om programmering i klasserommet og til slutt om dybdelæring i matematikk.

I kapittel 6 avslutter jeg denne oppgaven med en oppsummering og konklusjon i tillegg til en vurdering av pedagogiske implikasjoner og et tilbakeblikk på hele min forskningsprosess.

2 Teori

Et mål for skolen er at elevene skal oppnå dybdelæring i matematikk (Kunnskapsdepartementet, 2016, 2020). Et annet mål er at elevene i større grad skal lære å bli produsenter av teknologiske løsninger og ikke bare forbrukere av dem (NOU 2013:2; Sanne et al., 2016). I denne oppgaven ønsker jeg å undersøke om disse målene kan nås samtidig. Kan programmering integreres i matematikkfaget på en måte som gir begge målene et løft?

I dette kapitlet presenterer jeg teorien som ligger til grunn for forskningen jeg skal gjennomføre. Først prøver jeg å forstå begrepet dybdelæring, deretter hva dette vil si i matematikk. Jeg presenterer så hva som menes med programmering i denne oppgaven og viser til noe tidligere forskning på programmering i matematikk. Kapitlet avsluttes med en beskrivelse av fire fordeler en kan tenke med når en skal utarbeide rike undervisningsopplegg i matematikk hvor programmering står i fokus.

2.1 Dybdelæring

Da grunnlaget for dagens moderne skole ble til, virker det som om inspirasjonen var en blanding av et fengsel og en kirke. Elevene må møte til oppsatte tider, sitte i sine klasserom på rekke og rad, høre på læreren, og slipper ikke ut før timen er over. En visualisering av klasserommet hvor pultene står alene på rekke og rad er vanlig og individuelt arbeid har vært en mye brukt arbeidsmetode. Disse institusjonene ble formet *før* en hadde forsket på hvordan mennesker faktisk lærer (Sawyer, 2006). De som utformet skolene og dens innhold gjorde noen antakelser om læring, men som ikke var blitt vitenskapelig bevist. En antok at:

- Kunnskap er en samling fakta om verden og prosedyrer for hvordan problemer kan løses.
- Målet med skolen er å overføre disse faktaene og prosedyrene inn i hodene på elevene.
- Lærerne kan disse faktaene og prosedyrene og skal sørge for overføringen.
- En begynner med de enkleste faktaene og prosedyrene og øker kompleksiteten etter hvert.
- Måten en måler om en har lykkes med skolegangen er ved å teste hvor mange av disse faktaene og prosedyrene elevene har tilegnet seg.

(Sawyer, 2006, s. 1)

Dette tradisjonelle synet på skole er kjent som «instructionism» (Papert, 1993). Denne måten å tenke skole på fungerte muligens i en industrialisert verden, men verden i dag er blitt mye mer digitalisert. Den teknologiske utviklingen har vært eksponentiell og den tradisjonelle skolegangen klarer ikke lenger å forberede elevene på hvordan de skal delta i dette nye samfunnet (Sawyer, 2006, s. 1-2).

> «… memorization of facts and procedures is not enough for success. Educated graduates need a deep conceptual understanding of complex concepts, and the ability to work with them creatively to generate new ideas, new theories, new products, and new knowledge» (Sawyer, 2006, s. 2)

Hvis denne tradisjonelle måten å tenke skole på ikke fører fram til de målene vi har satt oss, må vi tenke annerledes. Debatten om skolen og dens innhold har ført til et større fokus på å forske på *hvordan* vi lærer. En studie av Marton og Säljö (1976) har fått mye oppmerksomhet, og begrepet dybdelæring blir beskrevet her. I studien var de interessert i å undersøke *hvordan* studenter og elever lærer, i stedet for *hvor mye* de kan lære. Et utvalg av studenter i høyere utdanning ble bedt om å lese en tekst og senere fortelle hva teksten handlet om. De ble også intervjuet om hvordan de gjennomførte lesingen.

Gjennom analyser av bidragene fant forskerne et tydelig mønster: Studentenes læringsprosesser var forskjellige og kunne kategoriseres i to grupper. «Den ene gruppen studenter pugget detaljert kunnskap for å stå til eksamen, mens den andre gruppen forsøkte å sette teksten de leste inn i en større sammenheng» (Gilje et al., 2018, s. 22). I studien ble de to læringsstrategiene betegnet som *surface level-processing (overflatelæring)* og *deep level-processing (dybdelæring)*. Studentene som viste tegn til overflatelæring fokuserte oppmerksomheten sin på selve teksten de var blitt bedt om å lese. Målet var å kunne reprodusere så mye av teksten som mulig senere i en prøvesituasjon. Læring handler derfor om å memorere og gjenkalle faktakunnskap. For elevene som viste tegn til dybdelæring ble oppmerksomheten rettet mot formålet med teksten: hva var det forfatteren av teksten ønsket å formidle? Studentene forsøkte å forstå teksten og sette den inn i en større meningsfull faglig sammenheng. Samtidig følte de seg aktive i læringsprosessen og brukte sin egen logiske sans til å konstruere kunnskap. Resultatet av senere utspørring viste at studentene med dybdelæring som læringsstrategi kunne gjengi mest (Marton & Säljö, 1976). Denne forskningen var et av flere utgangspunkt for å begynne å fokusere på forskjellen mellom dybdelæring og overflatelæring. Ludvigsen-utvalget (NOU 2015:8) fikk i oppdrag å vurdere grunnopplæringen i Norge opp mot krav om fremtidige kompetansebehov. De kom blant annet fram til at det må mer dybdelæring inn i skolen, og begrepet dybdelæring er nå blitt en sentral del av Fagfornyelsen, LK20.

Forskjellen mellom dybdelæring og overflatelæring kan vises forenklet i denne tabellen.

Dybdelæring	Overflatelæring
Elever relaterer nye ideer og begreper til tidligere kunnskap og erfaringer.	Elever jobber med nytt lærestoff uten å relatere det til hva de kan fra før.
Elever organiserer egen kunnskap i begrepssystemer som henger sammen.	Elever behandler lærestoff som atskilte kunnskapselementer.
Elever ser etter mønstre og underliggende prinsipper.	Elever memorerer fakta og utfører prosedyrer uten å forstå hvordan eller hvorfor.
Elever vurderer nye ideer og knytter dem til konklusjoner.	Elever har vanskelig for å forstå nye ideer som er forskjellige fra dem de har møtt i læreboka.
Elever forstår hvordan kunnskap blir til gjennom dialog og vurderer logikken i et argument kritisk.	Elever behandler fakta og prosedyrer som statisk kunnskap, overført fra en allvitende autoritet.
Elever reflekterer over sin egen forståelse og sin egen læringsprosess.	Elever memorerer uten å reflektere over formålet eller over egne læringsstrategier.

Tabell 1: Dybde- og overflatelæring *Kilde: Sawyer 2006, oversatt av Ludvigsen-utvalget (NOU 2014:7)*

Dybdelæring kan tolkes i lys av både kognitive- og sosiokulturelle læringsperspektiver (Gilje et al., 2018). Uavhengig av hvilket perspektiv en støtter seg til, vil fokus på kjerneelementene i fagene være viktig for å lykkes med dybdelæringen. Fra et kognitivt læringsperspektiv vil dybdelæring forutsette at de fakta som skal læres og det fagstoffet som skal forstås, settes inn i en relevant og forståelig sammenheng (National Research Council, 2000). «Med en relevant sammenheng vil det si at fagstoffet som skal læres, må knyttes til fagenes kjerneelementer» (Gilje et al., 2018, s. 24). Elevene må kjenne til fagets egenart og kjerneelementene som jo beskriver sentrale tenkemåter, metoder, prinsipper og begreper som utgjør kjernen i faget. Hvis fagstoffet knyttes til kjernen i faget, kan elevene se at det er relevant og forutsetningen for dybdelæring er til stede. Hvis fagstoffet ikke gjøres faglig relevant risikerer en at elevene opplever fagstoffet som fragmentert og det kan bli vanskelig å se sammenhenger.

Mens et kognitivt læringsperspektiv fokuserer på hvordan et individ tilegner og bygger egen kunnskap, vil et sosiokulturelt perspektiv beskrive «læring som prosesser og produkter der vi må se kognisjon og sosial samhandling i sammenheng» (Gilje et al., 2018, s. 24).

> «I det sosiokulturelle perspektivet er kjerneelementene viktige i det læringsarbeidet læreren legger opp til i klasserommet. Kjerneelementene kan anses som felles verktøy til å skape en forståelse av utvalgt og prioritert fagkunnskap. Kvaliteten i elevenes bidrag og utviklingen av læringsmiljøet er derfor sentrale elementer for at dybdelæring skal finne sted i arbeid med kjerneelementene" (Gilje et al., 2018, s. 24).

Dybdelæring kan ikke alene oppnås gjennom endringer i de kognitive funksjonene til individene. Det er også kvaliteten i samhandlingene mellom lærer og elev, og elevene i mellom, som påvirker hvordan læring skjer. Kjerneelementene i fag er igjen viktige, da de kan være med på å begrunne hvilket fagstoff som blir prioritert og sette føringer for utviklingen av læringsmiljøet.

I utformingen av norske utdanningspolitiske dokumenter, har man støttet seg til forskningsoppsummeringer som sammenfatter de viktigste forskningsresultatene innenfor kognitive og sosiokulturelle perspektiver på hvordan elever lærer og hva som kjennetegner god opplæring (Gilje et al., 2018; NOU 2014:7). Gjennom arbeidet med fagfornyelsen har «definisjonen» på dybdelæring hatt tre ordlyder:

> «Dybdelæring handler om at elevene gradvis utvikler sin forståelse av begreper og sammenhenger innenfor et fagområde. Det handler også om å forstå temaer og problemstillinger som går på tvers av fag- eller kunnskapsområder. Dybdelæring inn[e]bærer at elevene bruker sin evne til å analysere, løse problemer og reflektere over egen læring til å konstruere helhetlig og varig forståelse» (NOU 2014:7, s. 35), (NOU 2015:8, s. 14).
>
> «Dybdelæring betyr at elevene gradvis og over tid utvikler sin forståelse av begreper og sammenhenger innenfor et fag. Elevenes læringsutbytte øker når de gjennom dybdelæring utvikler en helhetlig forståelse av fag og ser sammenhengen mellom fag, samt greier å anvende det de har lært, til å løse problemer og oppgaver i nye sammenhenger» (Kunnskapsdepartementet, 2016, s. 14).
>
> «Vi definerer dybdelæring som det å gradvis utvikle kunnskap og varig forståelse av begreper, metoder og sammenhenger i fag og mellom fagområder. Det innebærer at vi reflekterer over egen læring og bruker det vi har lært på ulike måter i kjente og

ukjente situasjoner, alene eller sammen med andre» (Kunnskapsdepartementet, 2018b, s. 9).

Disse definisjonene uttrykker i stor grad det samme, men i siste delsetning av den tredje, og gjeldende, definisjonen for videre arbeid med læreplanene blir påvirkningen av det sosiokulturelle læringsperspektivet tydelig. Læring skjer når elevens individuelle kognitive utvikling kobles sammen med det sosiale samspillet i læringsmiljøet den er en del av.

Våre utdanningspolitiske dokumenter er tydelig påvirket av det sosiokulturelle læringsperspektivet, likevel sitter visualiseringen av et klasserom med pultene stilt enkeltvis på rekke og rad dypt i oss. Nå som stadig flere skoler klarer å tilby en digital enhet til hver elev (en-til-en klasserommet) endres undervisningspraksisen i klasserommet og her etterlyses mer forskning på hva som er god undervisningspraksis. *Gode eksempler på praksis (GEPP)* var et prosjekt som forsket på nettopp dette. Der kommer det fram at individuelt arbeid er en mye brukt og dominerende arbeidsform (Gilje et al., 2020). Skal vi lykkes med dybdelæring må det legges mer til rette for samhandling med andre i klasserommet.

Dybdelæring er viktig for å nå målene for skolen. Et mål med skolen er at elevene skal tilegne seg lærdom som gjør dem bedre rustet for å mestre, skape og leve et meningsfylt liv i samfunn og arbeid. Elevene må derfor «tilegne seg dyp forståelse av fagenes kjerneelementer som muliggjør refleksjon, problemløsning og idéskaping i nye situasjoner og på tvers av kunnskapsområder» (Gilje et al., 2018, s. 27).

2.2 Dybdelæring i matematikk

I forrige delkapittel pekte jeg på hvordan dybdelæring kan oppnås gjennom fokus på kjerneelementene i fagene. Her skal jeg se mer spesifikt på hva dybdelæring i matematikk kan være. Jeg presenterer først et mye brukt rammeverk for utvikling av matematikkompetanse. Deretter beskriver jeg kjerneelementene i matematikk og hvordan jeg kommer til å definere dybdelæring i matematikk i denne oppgaven.

Et kjent rammeverk for å beskrive kompetanse i matematikk er trådmodellen for matematisk kompetanse (Kilpatrick et al., 2001). Her beskrives komponentene i matematikklæring som tråder som flettes sammen til et tau. Poenget er at alle trådene er viktige for å tilegne seg kompetanse i matematikk. Matematikksenteret har bygget videre på denne og laget et norsk rammeverk som kan benyttes for å beskrive dybdelæring i matematikk (Nosrati & Wæge, 2018). Her trekkes det fram fem sentrale komponenter i den matematiske læringsprosessen for å beskrive hva dybdelæring i matematikk kan være. De har hentet og satt sammen sine komponenter av forskjellige forskningsbaserte og praksisnære modeller for læring i matematikk. Komponentene er i stor grad basert på Kilpatrick og hans kollegers trådmodell for matematisk kompetanse (Kilpatrick et al., 2001), men inkluderer også beskrivelser av relasjonell og instrumentell forståelse (Skemp, 1976), begrepsmessig og prosedyremessig kunnskap (Hibert & Lefevre, 1986), samt mange års forskningsresultater om metakognisjon og selvregulering i den matematiske læringsprosessen (Flavell, 1979; Schneider & Artelt, 2010).

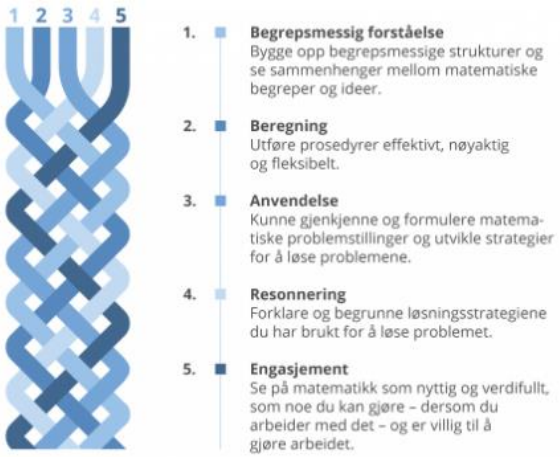
De fem komponentene er:

- **Begrepsmessig forståelse**
  Elevene skal «bygge opp begrepsmessige strukturer og se sammenhenger mellom ulike begreper, ideer og prosedyrer» (Nosrati & Wæge, 2018, s. 4). Det handler om å kunne mer enn isolerte regler og prosedyrer. Elevene skal se sammenhenger og relatere nye matematiske ideer til det de kan fra før. De rike relasjonene som oppstår vil også hjelpe elevene med å huske. Hvis en forstår hvorfor en metode fungerer, vil en også lettere kunne gjenskape metoden senere (Kilpatrick et al., 2001, s. 118). Med en begrepsmessig forståelse vil elevene kunne velge og bruke ulike representasjoner som er nyttige i ulike situasjoner.
- **Prosedyrekunnskap (Beregning)**
  Elevene skal «ha kunnskap om ulike matematiske prosedyrer og […] kunne utføre dem nøyaktig, fleksibelt og hensiktsmessig» (Nosrati & Wæge, 2018, s. 4). Ulike prosedyrer kan være viktige i ulike sammenhenger, elevene bør kunne velge hva som er hensiktsmessig i en gitt situasjon. Det er viktig at elevene ikke bare kan utføre prosedyren, men også forstå hvorfor de gjennomfører den og hvorfor den er gyldig. I tilknytning til prosedyrekunnskap finner vi også metoder for å estimere verdier som kan være til hjelp f.eks. når en skal vurdere gyldigheten til et svar. Algoritmer er prosedyrer. Elevene må se at det kan utvikles algoritmer som løser mange problem av samme type. Ved å studere algoritmene som «generelle prosedyrer» får elevene innsikt i at matematikk er veldig strukturert, forutsigbart og gjennomsyret av mønster (Kilpatrick et al., 2001, s. 118).
- **Anvendelse**
  Elevene skal «kunne gjenkjenne og formulere matematiske problemer, representere dem på ulike vis, utvikle en løsningsstrategi og vurdere hvor rimelig en løsning er» (Nosrati & Wæge, 2018, s. 5). Innenfor denne komponenten snakker vi også om strategisk- og problemløsingskompetanse. I situasjoner fra hverdagen kan det oppstå problemer som ikke er klart avgrenset og formulert. Elevene må kunne formulere problemet slik at de kan anvende matematikk til å løse det. Matematiske representasjoner og modeller er ofte en del av problemløsingsfasen (Kilpatrick et al., 2001, s. 118).
- **Resonnering**
  Elevene skal kunne forklare tankegangen sin og følge med på andres resonnement. De skal «kunne se og begrunne sammenhenger mellom ulike begreper, egenskaper og framgangsmåter» (Nosrati & Wæge, 2018, s. 6). Resonnering er selve limet som holder alt sammen, ledestjernen som viser veien fram til kompetanse. Begreper, prosedyrer og løsningsmetoder henger alle sammen på et vis, de gir mening (Kilpatrick et al., 2001, s. 118).
- **Metakognisjon og selvregulering (Engasjement)**
  Elevene skal kunne reflektere over hva de har lært, hvordan de lærer og hvorfor de lærer. «Når en elev begynner å bli bevisst sine egne læringsprosesser og strategier, står han også i en god posisjon til å gå inn og regulere dem» (Nosrati & Wæge, 2018, s. 6). Hvis elever skal utvikle de fire komponentene som er nevnt over, må de være overbevist om at matematikk er forståelig og gir mening, at de er i stand til å finne ut av det, at matematiske problemer kan løses ved hardt

arbeid og utholdenhet og at det er verdt det å bli god i matematikk (Kilpatrick et al., 2001, s. 118). Her synes jeg også det passer å vise til Carol Dweck (2012) sin forskning om tankesett. Hun viser til to typer grunnleggende tankesett som vi mennesker innehar. Mennesker med et *statisk tankesett* tror at alle våre egenskaper og evner er fastlåste og ikke kan endres i stor grad: Vi er født med en viss mengde intelligens og må forholde oss til denne. Mennesker med et *dynamisk tankesett* tror at våre grunnleggende kvaliteter kan endres. Det er mulig å utvikle egenskaper og evner gjennom hardt arbeid. Boaler (2016) viser til flere forskningsprosjekter hvor elevens tankesett er utslagsgivende på læringsutbyttet. Elever med et statisk tankesett gir lettere opp, mens elever med et dynamisk tankesett fortsetter å arbeide selv om oppgavene er vanskelige (Boaler, 2016, s. 6).

Disse fem komponentene er noe av forskningsgrunnlaget som ligger til grunn for utviklingen av kjerneelementene.

> «De fem komponentene i trådmodellen er tett bundet sammen og de støtter hverandre. Ved å arbeide med alle fem komponentene, vil elevene utvikle en solid, varig, fleksibel og relevant kompetanse i matematikk. De fem komponentene er i tråd med, og omfatter alle kjerneelementene i læreplanen, LK20» (Matematikksenteret, 2020).



Figur 1: "Trådmodellen"(Matematikksenteret, 2020)

Kjerneelementene i matematikk er:

- **Utforsking og problemløsing**
  Elevene skal lete etter mønster, finne sammenhenger og diskutere seg fram til en felles forståelse. Strategiene og fremgangsmåtene er viktigere enn løsningene. Elevene skal utvikle metoder for å løse problemer de ikke kjenner fra før. Algoritmisk tenking er viktig i prosessen med å utvikle strategier og fremgangsmåter.
- **Modellering og anvendelser**
  Elevene skal lage modeller som beskriver dagliglivet, arbeidslivet og samfunnet ellers. De skal få innsikt i hvordan de kan bruke matematikk i ulike situasjoner, både i og utenfor faget.
- **Resonnering og argumentasjon**
  Elevene skal kunne følge, vurdere og forstå matematiske tankerekker. Fremgangsmåter, resonnement og løsninger skal begrunnes.
- **Representasjon og kommunikasjon**
  Elevene skal kunne uttrykke matematiske begrep, sammenhenger og problem ved hjelp av representasjoner som kan være konkrete, kontekstuelle, visuelle, verbale og symbolske. Elevene skal kunne bruke et matematisk språk i samtaler, argumentasjon og resonnement.
- **Abstraksjon og generalisering**

Elevene skal kunne gradvis utvikle en formalisering av tanker, strategier og matematisk språk. Elevene skal kunne utforske tall, utregninger og figurer for å finne sammenhenger og deretter formalisere ved å bruke algebra og formålstjenlige representasjoner.

- **Matematiske kunnskapsområder**
  Elevene skal tilegne seg kunnskap om tall og tallforståelse, algebra, funksjoner, geometri, statistikk og sannsynlighet.

(Utdanningsdirektoratet, 2019b)

Den siste komponenten fra trådmodellen (Figur 1) til Matematikksenteret om metakognisjon og selvregulering (engasjement) finner man ikke igjen i kjerneelementene til faget, men en finner det igjen i overordnet del av læreplanverket, spesielt i kapittel 2.4 Å lære å lære (Kunnskapsdepartementet, 2020).

I klasserommet er det arbeidsmetodene og tenkemåtene beskrevet i kjerneelementene en ønsker at elevene skal engasjere seg i. Tankene bak de fem første kjerneelementene fikk gjennomgående stor tilslutning i høringsinnspillene (Utdanningsdirektoratet, 2017c) og anerkjennes da av fagmiljøene rundt i landet. Da kjerneelementene først ble offentliggjort i juni 2018 stod dette: «De fem første kjerneelementene beskriver arbeidsmåter, metoder og tenkemåter i matematikk. Det sjette kjerneelementet beskriver de sentrale kunnskapsområdene i matematikk. Elevene skal møte det sjette kjerneelementet gjennom de fem første kjerneelementene» (Kunnskapsdepartementet, 2018a, s. 15). Senere har den formuleringen forsvunnet, men jeg antar at det fortsatt er intensjonen med kjerneelementene.

Forenklet kan en si at hvis en bruker arbeidsmåtene, metodene og tenkemåtene som ligger i de fem første kjerneelementene til å tilegne seg fagstoff som er beskrevet i det sjette kjerneelementet, kan en oppnå dybdelæring i matematikk.

2.3 Programmering

Med programmering menes det i denne oppgaven en prosess (algoritmisk tenking) som leder fram til et produkt (algoritmer). I dette delkapitlet skal jeg først forklare noe av grunnen til at programmering nå skal inn i skoleverket. Deretter skal jeg beskrive mer i detalj hva programmering går ut på og knytte det opp mot *den algoritmiske tenkeren.* Videre presenterer jeg noe tidligere forskning på programmering i matematikk og beskriver hvilke programmeringsspråk jeg kommer til å bruke i mine undervisningsopplegg.

2.3.1 Programmering og matematikk

Digitale ferdigheter er blitt definert som en av fem grunnleggende ferdigheter i LK06. Men Digitutvalget (NOU 2013:2) pekte allikevel på den norske befolkningens manglende kompetanse i programmering som et hinder for digital verdiskaping i samfunnet. Digitale medier blir brukt til kommunikasjon, men det legges for liten vekt på å skape teknologi og allmennforståelsen av teknologisamfunnet er for dårlig. «For å sikre digital verdiskaping i fremtiden er vi nødt til å legge til rette for at barn og unge ikke kun er i stand til å bruke, men også skape digitalt innhold og digitale tjenester» (NOU 2013:2, s. 10). Vi kan ikke bare etterlate oss en oppvoksende generasjon av konsumenter.

Store deler av arbeidslivet er preget av den teknologiske utviklingen (Kunnskapsdepartementet, 2016, s. 6) og dermed må elevene få utvikle tilstrekkelig digital kompetanse mens de går på skolen hvis de skal være rustet for dette. «Teknologiutviklingen virker inn på alle fag, og digital kompetanse må komme til uttrykk i alle skolefagene» (NOU 2015:8, s. 10). I fremtiden vil mange av dagens elever være involvert i utviklingen av teknologi som er viktig for samfunnet og det har vært en økende trend i Europa å få integrert programmeringskompetanse i skolenes læreplaner (Balanskat & Engelhardt, 2015). To viktige grunner som legges fram er 1) det spås at i år 2020 vil Europa mangle over 800 000 arbeidstakere med utdanning innen informatikk, 2) programmeringskompetanse er viktig for å forstå nåtidens digitale samfunn og fremdyrke kompetanser for det 21. århundre som problemløsing, kreativitet og logisk tenking (Balanskat & Engelhardt, 2015, s. 6; Sanne et al., 2016).

Programmering er et ord som for mange skaper negative assosiasjoner. En tenker gjerne på en person som sitter isolert med kun sin datamaskin og skriver side opp og side ned med programkode (Bilde 1). Vedkommende utnytter gjerne svakheter i digitale systemer til egen vinning (hacker). Dette bildet tenker jeg skyldes uvitenhet og påvirkning fra film og TV. Jeg har hørt om en ansatt i Google som fortalte at hvis han kom inn til sjefen sin med ny programvare og påstod at han hadde skrevet det helt selv, uten hjelp fra andre, ville han trolig fått sparken! I samtale med en ansatt hos NRK som arbeider med utvikling av nettsiden (https://www.nrk.no/), ble jeg fortalt at i hans «team» var de 21 ansatte som alle arbeidet sammen, men med ulike ansvarsområder. Programmering handler ikke om å kun skrive kode på egenhånd.

Bilde 1: hentet fra B_A på Pixabay

> «Computer programming is the process of developing and implementing various sets of instructions to enable a computer to perform a certain task, solve problems, and provide human interactivity. These instructions (source codes which are written in a programming language) are considered computer programs and help the computer to operate smoothly» (Balanskat & Engelhardt, 2015, s. 6).

Programmering er en prosess hvor man utvikler og implementerer forskjellige instrukser som tillater at en datamaskin kan gjennomføre en oppgave, løse et problem eller samhandle med en bruker (Balanskat & Engelhardt, 2015, s. 7). Programmerere må kunne analysere, forstå og løse problemer ved å utvikle og verifisere algoritmer.

Algoritme er et annet ord som kan skape uheldige assosiasjoner. En algoritme er per definisjon «en fullstendig og nøyaktig beskrivelse av fremgangsmåten for løsning av en beregningsoppgave eller annen oppgave» (Hovde & Grønmo, 2020). For eksempel i matematikkopplæringen har de fleste elever blitt kjent med standardalgoritmer i de fire hovedregneartene (+, -, ·, :) som kan brukes til å løse beregningsoppgaver. Da stiller en gjerne opp tallene en skal regne med på bestemte måter og gjennomfører en innlært oppskrift som fører fram til et svar. Algoritmen angir de enkelte stegene i oppskriften og

rekkefølgen på dem. Alle dataprogrammer består av algoritmer som er skrevet i et programmeringsspråk. Det er disse algoritmene jeg kaller *produktet* av programmeringen.

I det første kjerneelementet i matematikk, *Utforsking og problemløsing*, skrives det om *algoritmisk tenking (AT)*.

> «Algoritmisk tenking er viktig i prosessen med å utvikle strategiar og framgangsmåtar for å løyse problem og inneber å bryte ned eit problem i delproblem som kan løysast systematisk. Vidare inneber det å vurdere om delproblema best kan løysast med eller utan digitale verktøy» (Utdanningsdirektoratet, 2020b).

AT er en noe uheldig norsk oversettelse av det engelske *Computational Thinking (CT)*. Det beskrives av eksperter som en grunnleggende ferdighet på lik linje med skriving, lesing og regning. «Computational thinking [CT] is a fundamental skill for everyone, not just for computer scientists … To reading, writing, and arithmetic, we should add CT to every child's analytical ability» (Wing, 2006). AT er mye mer enn bare algoritmer.

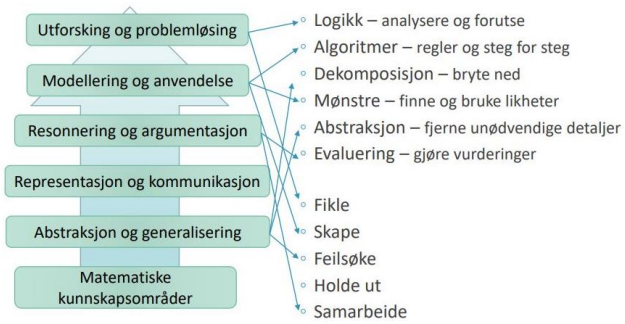
Algoritmisk tenking innebærer å tilnærme seg problemer på en systematisk måte. Komplekse problemer kan brytes ned til mer håndterlige delproblemer som kan løses først. Delproblemene kan fordeles blant de som samarbeider. Informasjon må organiseres og analyseres på en logisk måte før en kan pusle alt sammen til algoritmer som løser det opprinnelige problemet. AT er en problemløsningsmetode, men også en måte å uttrykke seg selv på gjennom digitale media. Tabell 2 viser noen viktige nøkkelbegrep og typiske arbeidsmåter den algoritmiske tenkeren bruker.

Nøkkelbegrep	Arbeidsmåter
Logikk - Analysere og forutse	Fikle – Utforske og eksperimentere
Algoritmer – Regler og steg-for-steg	Skape – Designe og lage
Dekomposisjon – Bryte ned i mindre deler	Feilsøke – Oppdage og rette feil
Mønstre – Finne og bruke likheter	Holde ut – Fortsette og prøve igjen
Abstraksjon – Fjerne unødvendige detaljer	Samarbeide – Dele og jobbe sammen
Evaluering – Gjøre vurderinger	

Tabell 2: Algoritmisk tenking *(Utdanningsdirektoratet, 2019a)*

Programmering setter i gang prosessene til den algoritmiske tenkeren. *Programmering* bruker jeg i denne oppgaven som en samlebetegnelse på både prosess og produkt, det er både algoritmisk tenking og algoritmer.

Algoritmisk tenking nevnes spesifikt i læreplanen for matematikk under kjerneelementet utforsking og problemløsing, men flere av begrepene ovenfor (Tabell 2) nevnes også i forbindelse med andre av kjerneelementene (Bilde 2). Programmering bør derfor være en glimrende måte å arbeide med matematikk i grunnskolen på, det treffer jo kjernen av faget.



Bilde 2: (Tofteberg, 2019)

Abstraksjon og generalisering er et av de nye kjerneelementene i matematikk. I følge Jeannette Wing (2008) er abstraksjon selve essensen i algoritmisk tenking. Algoritmer *er* abstraksjoner.

> «The most important and high-level thought process in computational thinking is the abstraction process. Abstraction is used in defining patterns, generalizing from specific instances, and parameterization. It is used to let one object stand for many. It is used to capture essential properties common to a set of objects while hiding irrelevant distinctions among them. For example, an algorithm is an abstraction of a process that takes inputs, executes a sequence of steps, and produces outputs to satisfy a desired goal» (Wing, 2011).

For å møte fremtidens behov for kompetanse beskriver Sanneutvalget (Sanne et al., 2016) algoritmisk tenking som det å kunne «abstrahere, gjennomgå informasjon systematisk, lære å lese og å forstå forskjellige representasjonsformer, modularisere [og] resonnere i iterative og parallelle strukturer» (s. 26). Også i denne beskrivelsen er det overlapp med kjerneelementene i matematikk og det er for meg naturlig at programmering skal inn i matematikkfaget.

2.3.2 LOGO-programmering

Programmering i matematikk er ikke noe nytt konsept. Da jeg begynte mitt arbeid med denne oppgaven ble jeg overrasket over å lese om Papert (1980) sine forsøk med LOGO. Mange av tankene mine hadde allerede blitt prøvd ut. Programmeringsspråket LOGO ble utviklet for å hjelpe barn med å lære programmering i et lekpreget miljø. Barn kunne skrive forholdsvis enkle koder og få en skilpadde til å bevege på seg. Skilpadden kunne være en fysisk robot plassert på gulvet, eller en markør på en dataskjerm. Etter hvert som skilpadden beveget på seg ville den etterlate spor, og dermed tegne linjestykker. Dette kunne utnyttes spesielt i undervisningen av geometri. Papert var en av utviklerne bak LOGO og mente at denne tilnærmingen til geometriundervisningen var bedre enn tilnærmingene til Euclid og Descartes, for dette kunne elevene kjenne seg igjen i. «Euclid's is a logical style. Descartes's is an algebraic style. Turtle geometry is a computational style of geometry» (Papert, 1980, s. 55). Skilpadden kunne relateres til elevenes egen fysiske kropp med både posisjon og retning. Gjennom LOGO kunne matematikkunnskapen utvikles ved å forstå og dirigere skilpaddens bevegelser. Elevene utvikler seg som problemløsere fordi de lærer seg å oppføre seg *som* matematikere, i motsetning til å kun lære *om* matematikk (Papert, 1972).

Det har blitt gjennomført en rekke forskningsstudier om LOGO-programmering i skolen hvor resultatene har variert (Clements et al., 2001; Clements & Sarama, 1995; Pea, 1987; Yelland, 1995). Clements og hans kollegaer var i stor grad positive til bruken av

LOGO i geometriundervisningen. Pea, derimot, klarte ikke å se noen forskjell i utviklingen av stadig mer komplekse kognitive ferdigheter hos barn som arbeidet med programmering og barna i kontrollgruppen som ikke arbeidet med programmering. I en nyere litteraturgjennomgang (Forsström & Kaufmann, 2018) vises det til at det ligger et stort potensial i å inkludere programmering som en del av matematikkundervisningen, men mer forskning etterspørres. For å lykkes, kreves nøye planlagte og gjennomtenkte undervisningsopplegg og gode veiledere med kunnskap om programmeringsspråket som er i bruk.

2.3.3 Programmeringsspråk

I denne oppgaven kommer jeg til å bruke programmeringsspråket Scratch (https://scratch.mit.edu/), som er et visuelt programmeringsspråk som er fritt tilgjengelig via internett. Scratch kan sees på som en videreutvikling av LOGO, hvor skilpadden er byttet ut med en katt (eller annen selvvalgt figur) og i stedet for å skrive koder med tekst kan programmereren pusle sammen ferdigprogrammerte blokker for å danne et program. Det kalles *blokkbasert* programmering.

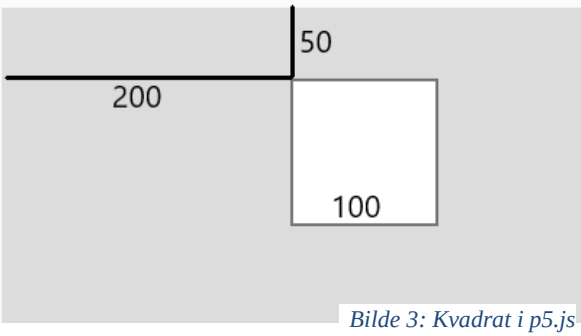
I Tabell 3 nedenfor viser jeg hvordan et tenkt program kunne sett ut i LOGO og Scratch. Begge programmeringsspråk støtter *løkker*, som er et grunnleggende programmeringsprinsipp. Ved å bruke en løkke kan du få programkoden som tilhører løkken til å gjenta seg. Dette er effektivt og kan spare deg for mye kodeskriving. I LOGO skriver man «repeat 4» for å fortelle programmet at koden som følger skal gjentas fire ganger. Kodene som skal gjentas må skrives etter innrykk. Skilpadden vil med denne koden gå fram 100 steg og vri seg 90 grader mot høyre. Dette vil gjentas fire ganger før programmet avsluttes. Resultatet blir at skilpadden beveger seg fra hjørne til hjørne i et kvadrat med sidelengde 100. Til høyre i tabellen ser vi et program i Scratch som gir akkurat det samme resultatet. Det visuelle brukergrensesnittet gjør at terskelen for å komme i gang senkes. I stedet for å bruke innrykk, ser en her visuelt at gjenta-blokken favner rundt gå- og snu-blokkene.

LOGO	**Scratch**
REPEAT 4 FORWARD 100 RIGHT 90 END	

Tabell 3: Forskjell på LOGO og Scratch

Etter noen år med fagfornyelsen vil forhåpentligvis programmeringskompetansen og interessen være økt hos elevene som kommer til ungdomsskolen fra barneskolen. Da vil det være hensiktsmessig å bevege seg videre fra blokkbaserte programmeringsspråk som Scratch, over til tekstbaserte programmeringsspråk. Jeg ønsker derfor å også undersøke hvordan elevene reagerer på et undervisningsopplegg i p5.js (https://p5js.org/) som også er fritt tilgjengelig via internett. p5.js er et JavaScript bibliotek som gjør det enkelt å tegne grafiske elementer på skjermen. Jeg tror elevene

har nytte av å se det visuelle resultatet av kodene de skriver så raskt som mulig og er derfor tilhenger av slike løsninger. I dette biblioteket finnes allerede mange funksjoner som vi kan bruke, for eksempel kan et kvadrat tegnes ved å skrive «square(200, 50, 100)». Et kvadrat tegnes med øvre, venstre hjørne på koordinatene (200, 50) og med sidelengde 100 (Bilde 3).



Bilde 3: Kvadrat i p5.js

2.4 Utforming av oppgaver

Programmering og algoritmisk tenking i matematikk klasserommet har blitt utprøvd i mange år, men uten at det har vært en del av læreplanene i ulike land. Store sprang i teknologiutviklingen og et mer digitalisert samfunn har nå resultert i at det blir integrert i læreplanen i stadig flere land og det kommer stadig nye lavterskels applikasjoner som gjør programmering tilgjengelig for alle alderstrinn.

Gadanidis og hans kollegaer (2017) har sammen med lærere utarbeidet rike undervisningsopplegg til bruk i skolen. Oppleggene var i starten ikke nødvendigvis knyttet til programmering, men etter å ha prøvd det ut synes de det styrker undervisningsmålene de har i matematikk og samtidig bidrar med nye muligheter. De har identifisert flere fordeler som kan assosieres med bruken av algoritmisk tenking i undervisningen. Fire av disse fordelene tenker de *med* når de skal planlegge økten. Disse fire fordelene er:

1) LIST (Lav inngangsterskel, stor takhøyde)
2) Konseptuell overraskelse
3) Vide vegger
4) Medvirkning

1) LIST innebærer at aktiviteten som velges til undervisningstimen har lav inngangsterskel, det vil si at det kreves minimalt med forkunnskaper for å komme i gang (Gadanidis et al., 2017). Aktiviteten er innbydende. Å spille *tre på rad* er et eksempel på et spill med lav inngangsterskel. *Sjakk*, derimot, krever en god del forkunnskaper og mange vil streve med å komme i gang. Hvis inngangsterskelen er for høy, kan flere elever bli frustrerte (Byrd, 2021).

Hvis aktiviteten i tillegg til en lav inngangsterskel har en stor takhøyde vil aktiviteten treffe et vidt spekter av elever som kan arbeide på sitt nivå. Aktiviteten skal oppleves som utfordrende, kreve anstrengelse og tillates å ta tid (Karlsen, 2014). Det ligger et utviklingsrom i oppgaven, et potensial, hvor elevene har mulighet til å utforske mer kompliserte ideer og representasjoner. Om taket i aktiviteten er høyt nok, kommer an på elevene. I *tre på rad* er det mulig å oppdage mønster for hvor en spiller bør plassere sine kryss. Hvis begge spillere spiller «korrekt» vil det alltid bli uavgjort, da er kanskje ikke taket så høyt. I sjakk derimot er det uendelig mange variasjoner og stadig noe nytt å lære.

2) Konseptuell overraskelse handler om å gi elevene mulighetene til å oppleve gleden av matematiske overraskelser. Det er ønskelig at elevene skal få en «aha-opplevelse». Ofte i det tradisjonelle matematikk klasserommet blir elevene presentert for

matematikken som andre har oppdaget. Her er det ønskelig at elevene skal gjøre sine egne oppdagelser. Det er mye å oppdage og utforske og mange følelser involvert. Overraskelsene kan gi ny matematisk innsikt. Når elevene kommer hjem til foreldrene sine og blir spurt om hva de gjorde på skolen i dag, ønsker vi at de skal ha en fortelling å dele fra matematikktimen (Gadanidis et al., 2017).

3) Vide vegger legger til en annen dimensjon til LIST. Programmering kan tilrettelegges for mange ulike prosjekter slik at elever med forskjellige interesser og læringsstiler alle kan bli engasjert. Med vide vegger kan elevenes matematiske tenking nå ut til et større publikum. Dette kan ses i sammenheng med konseptuelle overraskelser. Programmering kan øke mulighetene for at elevene opplever slike overraskelser og samtidig gi muligheter for å dele slike opplevelser med andre. Programmeringsspråkene Scratch og p5.js er begge nettbaserte og dermed lett tilgjengelige overalt og delbare. Elevene kan lagre programmer på nett og sende en lenke til et annet menneske som da får tilgang til hele programmet og kan gjøre egne endringer i koden. Ved å dele opplevelser får elevene muligheten til å gjenoppleve gleden av overraskelsene de har møtt gjennom øynene til familiemedlemmer eller andre (Gadanidis et al., 2017). Dette er en helt annen opplevelse enn å gjenfortelle hva læreren har sagt i en time.

4) Medvirkning handler om at det er eleven som skal ha kontrollen og være i sentrum av aktiviteten. Når elevene programmerer kan de skrive kode som gir mening for dem og de kan spore av fra den opprinnelige oppgaven for å undersøke andre problemer eller utvidelser. LIST miljøet gir muligheter for nettopp dette; utvide oppgaver og følge spor i selvbestemte retninger. Elevene er selv produsenter og ikke bare konsumenter og dette har betydning for holdningene elevene får til matematikk (Gadanidis et al., 2017). «I am convinced that the best learning takes place when the learner takes charge» (Papert, 1993, s. 25).

I min planlegging av undervisningsopplegg ønsker jeg å tenke med disse fire fordelene.

I dette kapitlet har jeg presentert teorien som ligger til grunn for forskningen jeg skal gjennomføre. Jeg har redegjort for hva som menes med dybdelæring i matematikk og hvordan begrepet programmering skal forstås i denne oppgaven. Videre har jeg kort presentert hvilke programmeringsspråk jeg skal benytte meg av i mine undervisningsopplegg og noen fordeler jeg ønsker å tenke med når jeg skal planlegge oppleggene.

I neste kapittel beskriver og begrunner jeg valg av forskningsdesign og metodene jeg har valgt å bruke. Jeg beskriver prinsipper for oppgavebaserte intervju og videre hvordan jeg foretok utvalget av elever til forskningen. Deretter presenterer jeg hvordan jeg har tenkt å analysere datainnsamlingen min. Kapitlet avsluttes med et blikk på etiske utfordringer og en vurdering av troverdigheten til denne oppgaven.

3 Metode

I dette kapitlet beskriver og begrunner jeg hvordan jeg går fram for å besvare forskningsspørsmålet mitt. Først skriver jeg om hvilke valg jeg tar angående forskningsdesign, metoder og hvordan dette er tenkt gjennomført. Deretter presenterer jeg prinsipper for oppgavebaserte intervju som blir hovedkilden min til datainnsamlingen. Videre redegjør jeg for hvordan jeg foretar utvalget mitt av elever og hvordan jeg kommer til å analysere datamaterialet. Til slutt skriver jeg om etiske utfordringer og om oppgavens troverdighet.

3.1 Valg av metode

I denne oppgaven ønsker jeg å undersøke hvordan programmering kan bli en integrert del av matematikkundervisningen på ungdomstrinnet. Jeg ønsker å undersøke om programmering som arbeidsmetode kan brukes for å lære matematikk slik det beskrives i læreplanen og om det bidrar til dybdelæring. Ettersom jeg introduserer en «ny» måte å arbeide på passer det å bruke en case-studie (Bell, 2005, s. 10). Med en case-studie får jeg muligheten til å gå i dybden på «problemet» programmering i matematikk og forske på hvordan elever på ungdomstrinnet responderer på å arbeide med programmering i matematikktimen. Deres erfaringer, opplevelser og utbytte vil danne grunnlaget for min analyse. «The basic case study entails the detailed and intensive analysis of a single case» (Bryman, 2016, s. 60). Jeg ønsker ikke at utvalget mitt i stor grad skal være det som avgjør mottakelsen av programmering og ønsker at utvalget skal være representativt, eller typisk, det Bryman (2016, s. 62) kaller eksemplifiserende. Derfor vil utvalget mitt bestå av tre caser. Da vil jeg ha muligheten til å undersøke ulike elevers respons og læring i arbeid med programmering. Jeg vil også kunne utvikle undervisningsopplegg som dekker flere fagområder og sammenligne de ulike oppleggene med tanke på hvordan programmeringen integreres i matematikkundervisningen.

Både kvantitative og kvalitative metoder kan brukes i case studier. Jeg ønsker å undersøke elevenes arbeidsprosesser og hvordan dette påvirker deres matematiske tenking, læring og forståelse. I tråd med den nye læreplanen i matematikk er det ikke svarene som er det mest interessante, men fremgangsmåtene og strategiene. Dermed passer det for meg å bruke en kvalitativ metode. «Qualitative research is a research strategy that usually emphasizes words rather than quantification in the collection and analysis of data» (Bryman, 2016, s. 364). De to mest brukte metodene i kvalitative studier er sannsynligvis deltaker-observasjon og intervjuer (Bryman, 2016, s. 492). Jeg kommer til å bruke begge metodene.

I forskningen min er jeg avhengig av at elevene arbeider med programmering i matematikkundervisningen. Jeg kan utvikle undervisningsopplegg og oppgaver som elevene møter i sine klasserom. Datainnsamlingen min kan da foregå ved at jeg observerer en hel klasse hvor deres egen matematikklærer gjennomfører opplegget jeg har laget. Dette alternativet byr på to hovedproblemer. For det første må læreren være godt kjent med opplegget og trygg på sine egne ferdigheter med det aktuelle programmeringsspråket i bruk. I tillegg vil det være svært vanskelig å få tilgang til elevenes arbeidsprosesser kun ved observasjon. «It is likely that there is a wide range of issues that are simply not amenable to observation, so that asking people about them represents the only viable means of finding out about them within a qualitative research

strategy» (Bryman, 2016, s. 494). Det første problemet kan løses ved at jeg deltar og blir medlærer i undervisningen. Da er jeg sikker på at opplegget introduseres som jeg har tenkt og elevene kan stille meg spørsmål hvis de støter på problemer. Det andre problemet kan løses ved at jeg får muligheten til å stille spørsmål til elevene for å få tilgang til deres tanker. Å være medlærer i undervisningstimen kan løse noen problemer, men også skape nye. Som medlærer vil jeg måtte være tilgjengelig for alle elevene og gå glipp av muligheten til å gå grundig inn i prosessene til én elev. Hvis jeg er opptatt et sted i klasserommet, vil jeg ikke kunne observere hva som skjer andre steder i klasserommet. Selv om jeg får det med på video- og lydopptak vil jeg ikke kunne stille spørsmål til elevene der og da. Å være medlærer og observatør samtidig vil være kaotisk.

Et annet alternativ som kan gi meg tilgang til elevenes tanker mens de arbeider er å gjennomføre oppgavebaserte intervjuer (Goldin, 2000). Et oppgavebasert intervju er et intervju der en eller flere oppgaver er en stor del av intervjuet og jeg som intervjuer kan følge med på oppgaveløsingen til eleven og stille oppfølgingsspørsmål ved behov. Dette kan jeg gjøre uten å involvere elevens lærer. Undervisningsopplegget kan gjennomføres i sin helhet uten resten av klassen. Ulempen her er at situasjonen blir mindre autentisk. Eleven er vanligvis en del av et større fellesskap hvor mulighetene for en-til-en undervisning er liten. Samspillet mellom elever vil også være noe jeg ønsker å observere og som jeg vil gå glipp av. I tillegg vet jeg at mange lærere ønsker å bli eksponert for programmering i klasserommet og vil sette pris på at undervisningsopplegget gjennomføres i hele klassen.

Med bakgrunn i de overstående avsnittene beslutter jeg å gjennomføre oppgavebaserte intervju med elevpar etter at jeg har introdusert undervisningstimen som medlærer. Jeg sikrer at introduksjonen til timen gjennomføres som jeg har tenkt, uten at klassens lærer trenger å bruke mye tid i forkant til forberedelse. Dette gjør det også lettere å få lærere til å melde seg og sin klasse som forsøkspersoner i forskningen min. Etter introduksjonen arbeider elevene i par med oppgavene jeg har laget. Ett par blir med meg ut av klasserommet og inn på et grupperom, hvor jeg gjennomfører et oppgavebasert intervju.

Med oppgavebaserte intervju skapes det et til dels kontrollert matematisk miljø. Sammenlignet med tradisjonelle papir-og-blyant prøvemetoder, er det med oppgavebaserte intervju mulig for forskeren å vektlegge elevens prosesser i arbeidet med matematiske oppgaver, istedenfor kun resultatene som produseres (Goldin, 2000). Disse intervjuene vil være semi-strukturerte hvor selve oppgavene fungerer som intervjuguiden. Det er viktig at elevene arbeider i par slik at jeg kan få tilgang til interaksjonen dem imellom for å bedre forstå deres prosesser. Både programmering og kjerneelementene i matematikk legger til rette for samarbeid. Hadde jeg kun intervjuet én elev ville jeg ikke kunnet svare på forskningsspørsmålet mitt. Samtidig er det noen andre utfordringer knyttet til intervju med par som ikke gjelder en-til-en intervjuer. Hvis den ene eleven har høyere status eller sterkere personlighet enn den andre, kan den styre for mye av arbeidet og gjøre det vanskelig for den andre å delta (Bell, 2005, s. 163). Jeg må derfor være oppmerksom på at begge elevene får uttrykt sine tanker. Ettersom arbeidet skal foregå på en datamaskin med et tastatur og en datamus, vil jeg be den ene eleven styre datamusen og den andre tastaturet i et forsøk på å involvere begge elevene i arbeidet.

Det blir spilt inn lyd og bilde fra intervjuene som jeg i etterkant kan analysere nærmere.

3.2 Prinsipper for oppgavebaserte intervju

Goldin (2000, s. 539-544) har, basert på sin egen erfaring, foreslått ti metodiske prinsipp for å designe og gjennomføre oppgavebaserte intervju. Jeg vil forsøke å etterfølge disse for å sikre kvaliteten på intervjuene mine.

«Design task-based interviews to address advance research questions.» Forskningsspørsmålene må være klare på forhånd og tydelig definerte, slik at intervjuet kan tilpasses til å besvare disse. Spørsmålene som skal besvares vil påvirke valg av oppgaver og verktøyene som tas i bruk. Mitt forskningsspørsmål er *hvordan kan programmering bli en integrert del av matematikkundervisningen på ungdomstrinnet?* Det jeg ser etter er tegn på at tanke- og arbeidsmåtene til den algoritmiske tenkeren bidrar til å utvikle kompetanser beskrevet i kjerneelementene i matematikk. Undervisningsoppleggene jeg utvikler vil forhåpentligvis skape en setting hvor jeg kan få svar på dette.

«Choose tasks that are accessible to the subjects.» Oppgavene må være tilpasset elevene som skal intervjues. Jeg vil legge vekt på å utvikle LIST-oppgaver slik at alle kan få til noe, uavhengig av hvordan de ellers klarer seg i matematikkfaget. Programmering vil sannsynligvis være nytt for de fleste og oppleggene jeg lager vil være tilrettelagt for dette. Samtidig ønsker jeg å ha en progresjon i vanskelighetsgrad, slik at alle elever vil få noe å bryne seg på.

«Choose tasks that embody rich representational structures.» Oppgavene bør være rike; gi rom for forskjellige måter å tenke på og ulike representasjoner. Det er ønskelig med oppgaver som inneholder matematiske strukturer som det er mulig å abstrahere. Abstraksjon er noe av kjernen i programmering og også et av kjerneelementene i matematikk. Gjennom programmeringsspråkene jeg foreslår vil det også være mulig å visualisere matematikken og få umiddelbar tilbakemelding hvis programmet ikke oppfører seg som forventet. Utfordringen ligger i om elevene vil ha gode nok forutsetninger til å se mulighetene i oppgavene de får ettersom de sannsynligvis har begrensede erfaringer med programmering.

«Develop explicitly described interviews and establish criteria for major contingencies.» Så mange av detaljene som mulig må være på plass før intervjuet starter slik at andre forskere kan få innsyn i valgene som er blitt tatt og får muligheten til å diskutere, kopiere og videreutvikle forskningen. I mine intervju ønsker jeg å hjelpe så lite som mulig med oppgaveløsingen. Når det gjelder det tekniske, selve programmeringsspråkene, så vil jeg være mer behjelpelig. Jeg vil be elevene fortelle hva de forventer skal skje før de kjører et program de har laget. Da slipper jeg å stille det spørsmålet i ettertid, og det vil være enklere å forstå verbale og ikke-verbale uttrykk. De fleste spørsmålene jeg stiller vil være «oppklarende» spørsmål som «Hvorfor gjør du…?» og «Hva mener du når du sier…?».

«Encourage free problem solving.» Elevene må få tid til å selv forsøke å løse problemene uten at jeg blander meg inn. Da får jeg observert deres spontane reaksjoner og impulser. Dersom de står helt fast vil jeg forsøke å veilede dem videre. Det vil sannsynligvis være utfordrende for elevene å begi seg ut på fri problemløsing, ettersom programmering antakelig er nytt for de fleste elevene.

«Maximize interaction with the external learning environment.» Gjennom interaksjon med eksterne kilder, kan en få tilgang til elevenes interne prosesser. Oppgavene i disse intervjuene skal alle løses på datamaskin i et valgt programmeringsspråk. Elevene vil være i konstant samhandling med datamaskinen; mus, tastatur og skjerm. I tillegg vil de ha kladdeark og skrivesaker tilgjengelig.

«Decide what will be recorded and record as much of it as possible.» Forskningsspørsmålet og det man er på jakt etter å observere vil styre hva slags opptak man tar i bruk. Jeg kommer til å ta både lyd- og videoopptak. Video av elevene slik at jeg kan observere og tolke ansiktsuttrykk, og video av det som gjøres på skjermen slik at jeg kan tolke verbale og ikke-verbale uttrykk i lys av hva elevene arbeider med.

«Train the clinicians and pilot-test the interview.» De som skal utføre intervjuene må øve på å gjennomføre dem likt, og oppgavene bør testes ut på forhånd slik at uheldig språkbruk eller andre misforståelser kan oppdages og utbedres før oppgavene tas i bruk. Jeg er den eneste som skal utføre intervjuene, så det er ikke noe behov for å trene opp andre. På grunn av tidsbegrensing vil jeg ikke pilotere oppgavene på andre elever, men diskutere dem med andre kollegaer og ansatte på universitetet.

«Design to be alert to new or unforeseen possibilities.» Elever kan ofte overraske oss med sin tenking. Hvis vi er flinke til å lytte vil vi kanskje oppdage nye muligheter eller perspektiver som kan påvirke ideene våre. Jeg vil under intervjuene forsøke å lytte til elevenes tanker og være åpen for uforutsette tolkninger og innspill.

«Compromise when appropriate.» Under intervjuene kan det oppstå konflikter mellom prinsipper, for eksempel kan tidsbegrensing stå i veien for at elevene får nok tid til problemløsingen på egenhånd. Jeg er forberedt på at ikke alt vil gå som planlagt og at jeg må gjøre vurderinger og ta avgjørelser underveis for å sikre best mulig utgangspunkt for mine analyser.

3.3 Utvalg

Samtidig som jeg gjennomførte forskningen min arbeidet jeg på en ungdomsskole og var deltakende i et fagutviklingsnettverk i matematikk med andre skoler i området. Jeg henvendte meg til kollegaer via e-post og etterspurte en klasse hvor jeg kunne låne en time. I mailen skrev jeg forslag til hva faginnholdet i timen kunne være og hvordan jeg tenkte at timen skulle gjennomføres, men var samtidig åpen for å tilpasse mitt innhold til deres behov. Jeg presiserte at læreren selv ikke trengte å kjenne til det aktuelle programmeringsspråket på forhånd ettersom jeg selv kunne lede introduksjonen. Den aktuelle klassens timeplan måtte passe overens med min egen, dette løste seg overraskende greit.

Datainnsamlingen skulle i utgangspunktet foregå over en periode på 1,5 år. Første opplegg ble gjennomført våren 2019. Det andre opplegget ble gjennomført høsten 2019. Det tredje opplegget skulle etter planen gjennomføres våren 2020, men på grunn av COVID-19 utbruddet ble dette i første omgang utsatt til høsten 2020. Stadige endringer i forhold til smittevern på skolene førte til at opplegget ikke lot seg gjennomføre da heller. Derfor har jeg kun fått gjennomført to av mine tre planlagte undervisningsøkter som tenkt. Det siste undervisningsopplegget har jeg derimot fått prøvd ut i egen klasse uten å gjennomføre noe intervju. Det ble heller ikke tatt lyd- eller videoopptak.

Informasjonsrunden med alle klassene foregikk på samme måte. Etter at jeg hadde opprettet kontakt med en velvillig lærer besøkte jeg klassen 1-2 uker før jeg skulle gjennomføre datainnsamlingen. Jeg orienterte kort om hva forskningen min handlet om og etterspurte elever som kunne tenke seg å hjelpe meg i mitt arbeid. Jeg forklarte hvordan det ville foregå og delte ut informasjonsskrivet (8.1 Informasjonsskriv) til de som meldte seg.

I første runde med datainnsamling var det omtrent 10 av 18 elever i klassen som meldte sin interesse da jeg besøkte klassen første gang. Alle fikk informasjonsskrivet og ble minnet på å få underskrift flere ganger i løpet av den neste uken av læreren sin. Da jeg kom for å gjennomføre opplegget var det kun tre av elevene som hadde fått underskrift. Jeg trakk tilfeldig to av navnene, begge var jenter.

I andre runde med datainnsamling var det 4 av 17 elever som meldte seg frivillig. Disse fire fikk også påminnelser om å få underskrift på samtykkeerklæringen. Da jeg kom for å gjennomføre opplegget var det kun to av elevene som hadde fått underskrift, begge var gutter, og disse ble mine utvalgte.

I de to første klassene startet jeg som planlagt med alle elevene samlet. Deres lærer delte dem i par og de arbeidet på én maskin sammen. Etter introduksjonen min ble de to utvalgte elevene med meg ut av klasserommet og inn på et grupperom hvor jeg på forhånd hadde gjort klar en PC med eksternt tilkoblet tastatur, mus og 24" skjerm. Det ble også gjort lyd- og videoopptak. Elevene ble bedt om å tenke høyt så mye som mulig.

I min egen matematikklasse ble det tredje opplegget prøvd ut i en periode hvor skolen var på rødt smittevernsnivå. Det innebærer at det kun var halve klassen til stede på skolen hver dag. Jeg gjentok derfor det samme opplegget to ganger, en gang med hver halvdel, slik at det ble gjennomført med alle elevene. I disse to timene var jeg mer lærer enn forsker og det ble ikke gjennomført systematisk datainnsamling. Elevene ble gjort kjent med at dette var et egenutviklet undervisningsopplegg til masteroppgaven min om programmering i matematikk. Jeg forsikret dem om at jeg ikke kom til å samle inn noen personsensitive opplysninger eller bruke navn. Første halvdel bestod av åtte elever og ble delt inn i fire par. Andre gruppe bestod av ti elever og ble delt inn i fem par.

3.4 Hvordan materialet analyseres

I denne oppgaven ønsker jeg å bruke en kombinasjon av nøkkelbegrepene og arbeidsmåtene knyttet til den algoritmiske tenkeren og kjerneelementene i matematikk som mitt teoretiske rammeverk. Ettersom jeg ønsker å finne ut av hvordan programmering kan integreres i matematikkundervisningen, skal jeg først undersøke om elevene faktisk er engasjert i en programmeringsaktivitet og arbeider som algoritmiske tenkere. Deretter vil jeg vurdere om elevene også arbeider med kjerneelementene i matematikk.

Programmering har jeg i denne oppgaven definert som todelt; det handler om algoritmisk tenking (prosessene) og algoritmer (produktet). I min analyse kommer jeg til å først stadfeste om elevene arbeider som programmere. Til dette vil jeg bruke nøkkelbegrepene og arbeidsmåtene til den algoritmiske tenkeren (Tabell 4Tabell 4). Jeg kommer til å bruke beskrivelsene i tabellen som kjennetegn på når begrepene og arbeidsmåtene er i bruk. I tillegg vil jeg presentere utklipp av elevenes arbeid på dataskjermen. Da får leseren et innblikk i både elevenes prosesser og produkter.

	Begrep	Beskrivelse
Nøkkelbegrep	Logikk	Analysere og forutse
	Algoritmer	Regler og steg-for-steg
	Dekomposisjon	Bryte ned i mindre deler
	Mønstre	Finne og bruke likheter
	Abstraksjon	Fjerne unødvendige detaljer
	Evaluering	Gjøre vurderinger
Arbeidsmåter	Fikle	Utforske og eksperimentere
	Skape	Designe og lage
	Feilsøke	Oppdage og rette feil
	Holde ut	Fortsette og prøve igjen
	Samarbeide	Dele og jobbe sammen

Tabell 4: Den algoritmiske tenkeren

Siden denne oppgaven ikke bare handler om programmering, men programmering i matematikk på ungdomstrinnet, må elevenes arbeid også knyttes opp mot skolefaget matematikk. I andre del av analysen vil jeg derfor se om elevenes arbeid også kan knyttes til beskrivelsene av kjerneelementene i matematikk. I tillegg vil jeg forsøke å kode datamaterialet ved å knytte de enkelte utsagnene til elevene til kjerneelementene. Under et av masterseminarene i regi UiA våren 2019 fikk jeg se hvordan en medstudent laget en visuell oversikt over elevenes bruk av de ulike trådene i trådmodellen til Kilpatrick et al. (2001) i en oppgavesituasjon (Sletten, 2019). Denne visuelle modellen likte jeg svært godt og vil tilpasse den til mitt eget bruk. I stedet for å bruke kompetansebeskrivelsene i trådmodellen, bruker jeg kjerneelementene i matematikk som mitt utgangspunkt.

Kjerneelementene vil jeg operasjonalisere på følgende måte:

Utforsking og problemløsing	- Elevene leter etter mønster og sammenhenger - Elevene diskuterer seg fram til en felles forståelse - Elevene arbeider med problemer de ikke kjenner fra før - Elevene bryter ned et problem i delproblem - Elevene vurderer om løsningene er gyldige
Modellering og anvendelser	- Elevene lager modeller som beskriver noe fra virkeligheten - Elevene bruker sin matematiske kunnskap i nye situasjoner
Resonnering og argumentasjon	- Elevene utformer egne resonnement - Elevene forstår andres resonnement - Elevene begrunner fremgangsmåter
Representasjon og kommunikasjon	- Elevene representerer matematiske begrep, sammenhenger og problemer - Elevene bruker matematisk språk - Elevene veksler mellom ulike representasjoner
Abstraksjon og generalisering	- Elevene formaliserer tanker, strategier og/eller matematisk språk - Elevene oppdager sammenhenger og strukturer som de formaliserer ved hjelp av algebra og/eller andre hensiktsmessige representasjoner
Matematiske kunnskapsområder	- Elevene arbeider med fagstoff fra de matematiske kunnskapsområdene

Tabell 5: Operasjonalisering av kjerneelementene

Jeg skal planlegge tre undervisningsøkter i matematikk med programmering som hovedaktivitet. Jeg vil etter innhenting av data sitte med to (opprinnelig plan var tre) oppgavebaserte intervjuer som hver og en varer i underkant av en time. Det blir gjort video- og lydopptak av intervjuene. I tillegg vil jeg ha egne tanker og refleksjoner rundt oppleggene og elevenes utbytte. Analysen av materialet vil hovedsakelig foregå gjennom fire steg.

3.4.1 Feltnotater

Første steg i analysen vil være mine egne feltnotater fra introduksjon av timen, intervjuene og inntrykkene jeg sitter igjen med etter at timen er over. Jeg har ingen strukturert plan med feltnotatene og ønsker ikke å notere ned mye, da dette vil stjele oppmerksomheten min vekk fra elevene. Målet med feltnotatene er å notere ned tanker og spørsmål som slår meg underveis slik at jeg ikke glemmer dem. I tillegg vil jeg notere ned tidspunkter under intervjuene der noe skjer som jeg tenker vil være nyttig å se om igjen på video. Spesifikke utsagn eller kroppsspråk som skjer i en kontekst det kan være vanskelig å fange opp på video vil jeg også notere ned. Dette vil nok hjelpe meg når jeg i ettertid skal transkribere intervjuene. Når intervjuet er avsluttet vil jeg også notere ned stikkord som kan hjelpe meg med analysen i ettertid. Feltnotatene blir ikke digitalisert.

3.4.2 Transkripsjon

En transkripsjon er en oversettelse fra muntlig språk til skriftspråk. I etterkant av intervjuene vil jeg så fort som mulig se gjennom opptakene og forsøke å transkribere nøyaktig hva som blir sagt (8.2 Transkriberingsnøkkel). Oversettelsen fra tale til skrift fanger ikke opp hele konteksten. Kommunikasjon er mer enn bare ord. Jeg vil derfor også beskrive ikke-verbale uttrykk og legge ved skjermutklipp av hva elevene arbeidet med da ordene blir sagt. Forhåpentligvis vil konteksten bli lettere tilgjengelig for leseren. Når jeg skriver ned det som er blitt sagt, vil jeg forsøke å skrive ordene med riktig staving, selv om de uttales på dialekt på annen måte, så lenge det ikke endrer innholdet i utsagnet. For eksempel vil jeg skrive «jeg», selv om det som blir sagt høres ut som «æ». Hovedformålet med dette første steget er å transkribere intervjuene, men samtidig som jeg gjennomgår datamaterialet vil jeg begynne å gjøre meg opp noen tanker rundt hvilke episoder jeg tenker kan spille en sentral rolle i min videre analyse.

3.4.3 Algoritmisk tenking

Etter å ha transkribert intervjuene har jeg gjennomgått elevenes oppgaveløsing i to omganger. Første gang direkte mens de oppgavebaserte intervjuene pågikk, og deretter en gang via video- og lydopptak. Da har jeg god oversikt over hva som har skjedd og vil plukke ut episoder som egner seg for gjengivelse. Hensikten med gjengivelsene vil være å vise til at elevene arbeider som programmere og bruker arbeidsmåter og nøkkelbegrep knyttet til den algoritmiske tenkeren (Tabell 4Tabell 4). I analysen min kommer jeg til å vektlegge disse arbeidsmåtene og begrepene og skrive dem i kursiv.

3.4.4 Kjerneelementene

Hvis jeg lykkes med programmeringsaktiviteten og elevene faktisk arbeider som programmere, kan jeg gå videre i min analyse og vurdere om elevene samtidig arbeider med matematikk slik det er beskrevet i læreplanen. Nå vil jeg gjennomgå transkripsjonen enda en gang og fargekode utsagnene som kan knyttes opp til et kjerneelement slik som beskrevet over (Tabell 5). Dersom utsagnet ikke kan knyttes til et kjerneelement, eller det er jeg selv som snakker vil utsagnet ikke få en fargekode, men allikevel være med i oversikten. Det er mange tenkepauser i transkripsjonen hvor

elevene for eksempel sier «hmm». Selv om elevene her tenker og sannsynligvis bedriver matematikk kommer jeg ikke til å fargekode slike utsagn. Det vil dermed bli en del utsagn uten fargekode. Hvis et utsagn kan knyttes til flere kjerneelement vil det få flere fargekoder. Noen episoder består av flere linjer transkripsjon hvor utsagnene hver for seg ikke nødvendigvis kan knyttes til et kjerneelement, men hvor helheten kan. Her gir jeg et eksempel:

Kari	[02:17] Jaa! (Hendene i været) Det ble det, det var bare det at vi glemte å slette alt.						
Trine	[02:24] Vi kan slette før …						
Kari	[02:25] Vi kan ta på sånn slett alt ting … Med den						
Trine	[02:29] Ja, slett alt. Før alt annet. Ja.						
Kari	[02:31] Og så må vi sette han på at han går til den derre … her? (Blokk: gå til)						
Kor	[02:38] Null, null.						

I de fem første utsagnene av denne episoden arbeider elevene med det første kjerneelementet utforsking og problemløsing: Elevene diskuterer seg fram til en felles forståelse mens de arbeider med et problem de ikke kjenner fra før. I det siste utsagnet står det bare «null, null». Men av konteksten kommer det fram at det er koordinatene (0 , 0) det er snakk om. Når elevene sier «han» sikter de til katten i Scratch. De snakker om å «slette alt» som har blitt tegnet før, dette må bli en del av algoritmen. De bryter da problemet ned i delproblem; først sletter de alt, deretter flytter de katten til koordinatene (0 , 0). Her representerer elevene koordinatene både skriftlig i koden sin ved hjelp av x- og y-verdier (de må skrive «gå til x: 0 y: 0») og visuelt ved hjelp av visningsvinduet i Scratch hvor katten flytter seg til den ønskede plasseringen. De bruker et matematisk språk og ulike representasjoner; dermed kan utsagnet knyttes til kjerneelementet representasjon og kommunikasjon. Elevene arbeider med geometri i et koordinatsystem og arbeider derfor også med det siste kjerneelementet: matematiske kunnskapsområder.

3.5 Etiske utfordringer/vurderinger

En forskningsoppgave som denne er et stort stykke arbeid som involverer flere mennesker enn bare meg selv. Det er viktig å tenke over etiske utfordringer som kan oppstå og sørge for at vurderingene man tar er i tråd med etiske retningslinjer. Hva som er rett og galt kan diskuteres. Et viktig kriterium er å sørge for at forskningen som gjennomføres samsvarer med ens egne etiske prinsipper (Bell, 2005).

Bell (2005) har skrevet en sjekkpunktliste (s. 57-58) for etiske vurderinger som jeg tar utgangspunkt i når jeg skriver de neste avsnittene.

I mitt arbeid intervjuer jeg flere ungdommer og ønsker å si noe om deres tenkemåter. Jeg tar lyd- og videoopptak og har til hensikt å publisere mine funn. Dette må tenkes grundig gjennom for å ivareta rettighetene til mine informanter og konsekvensene det kan få for dem. NSD – Norsk senter for forskningsdata AS har vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket. Elevenes klasser har jeg oppsøkt 1-2 uker i forveien for å muntlig fortelle om og beskrive prosjektet. Det har vært frivillig for elevene å delta. Elevene som var

interessert fikk med seg hjem et samtykkeskjema hvor jeg informerte på nytt om hva prosjektet handler om, hvor de kan få mer informasjon og at det var frivillig å delta. Elevene fikk også beskjed, både muntlig og skriftlig, at de når som helst kunne trekke seg fra prosjektet og at dette ikke ville få noen negative konsekvenser for dem. Jeg har forsøkt å sikre deltakernes konfidensialitet; jeg bruker ikke deres ekte navn og oppgir kun kjønn og at de er elever på en ungdomsskole. Alle lyd- og videoopptak er det kun jeg som har hatt tilgang til og disse slettes så snart denne oppgaven er vurdert og bestått.

Utvalget mitt av elever har jeg fått via lærere jeg allerede kjente fra før. Jeg har ikke ønsket å forske på mine egne matematikkelever, men har gjort det til opplegg 3 på grunn av begrunnelser allerede nevnt. Elevene i de andre klassene var ikke mine matematikkelever, men jeg hadde vært innom begge klassene tidligere som vikarlærer i en time. Elevene visste hvem jeg var og det kan ha påvirket dem. Dette trenger ikke nødvendigvis å ha vært noe negativt, det kan ha gjort intervjusituasjonen enklere ettersom jeg ikke var en totalt fremmed. Men det kan også ha skapt en skjev maktbalanse som jeg har forsøkt å rette på. Elevene fikk tydelig beskjed om at deltakelsen og intervjuet ikke ville ha direkte innvirkning på vurderingen deres i faget, annet enn muligheten for å lære mer. Deltakelsen ville heller ikke påvirke lærerens eller skolens forhold til dem. Håpet er at prosjektet vil bidra til å belyse et aktuelt didaktisk tema og deres deltakelse kan være med på å bedre undervisningspraksis og dermed tjene både elever og lærere på sikt.

Forskningen skal si noe om matematikkundervisning som vanligvis foregår i et klasserom, derfor oppstår en unaturlig situasjon for elevene som skal ut på et grupperom for å bli intervjuet med lyd- og videoopptak. De oppfører seg sannsynligvis annerledes enn vanlig, spesielt siden jeg ber dem tenke høyt. Det kan virke litt søkt å skulle si noe om programmering i matematikklasserommet når jeg baserer funnene mine på elever i par på et grupperom, men med tidsbegrensingen og mulighetsrommet jeg hadde mener jeg dette allikevel er et godt valg.

Å være lærer og forsker samtidig kan være krevende. Jeg skriver om et tema jeg er engasjert i og har meninger om, men må forsøke å holde det på sidelinjen og må i analysen og konklusjonen kun skrive om det jeg har belegg for å skrive noe om.

3.6 Troverdighet

Det er vanlig å vurdere kvaliteten av forskning ved hjelp av to kriterier: Reliabilitet og validitet. Reliabilitet beskriver i hvor stor grad forskningen kan gjentas med samme utfall, «om gjentatte målinger med samme måleinstrument gir samme resultat» (Ringdal, 2013, s. 96). Validitet, eller gyldighet, beskriver i hvor stor grad det er sammenheng mellom det det forskes på og konklusjonene, måler vi det vi ønsker å måle? «[…] validity refers to whether you are observing, identifying, or measuring what you say you are» (Bryman, 2016, s. 383).

Reliabilitet og validitet er mye brukt til å måle kvaliteten av kvantitativ forskning. Begrepsinnholdet må tilpasses for å passe til kvalitativ forskning. I kvantitativ forskning er en ofte i stor grad opptatt av målbare resultater, men i min kvalitative forskning er «resultatene» en konsekvens av en sosial setting med samhandling mellom flere mennesker og enheter. Hvordan elevene forstår og løser oppgavene vil i stor grad

preges av deres egne forutsetninger, erfaringer og holdninger og ikke bare den felles introduksjonen til den aktuelle timen som jeg har ansvaret for. Dermed vil ikke forskningen min nøyaktig kunne gjenskapes.

Bryman (2016) foreslår derfor et alternativ til kriterier for å vurdere kvalitativ forskning (s. 384). Jeg kommer til å bruke fire kriterier for å vurdere kvaliteten av forskningen min som samlet kan kalles troverdighet. De fire kriteriene er *kredibilitet*, *overførbarhet*, *pålitelighet* og *bekreftbarhet*.

3.6.1 Kredibilitet (Credibility)

Den sosiale verden kan tolkes ulikt fra person til person. Å påstå at en har observert en objektiv sannhet vil være en tvilsom påstand. For å sikre troverdighet i forskning gjennomført i sosiale situasjoner må en sørge for at forskningen gjennomføres i tråd med føringer for god praksis og at funnene legges fram for deltakerne i forskningen for å sikre at deres sosiale verden er blitt riktig forstått (Bryman, 2016, s. 384).

Denne oppgaven håper jeg tilfredsstiller kravene til god forskningspraksis. Jeg har forholdt meg til retningslinjene og veiledningen jeg har fått. Jeg har søkt og fått godkjent tillatelse fra NSD til å innhente informasjon fra elevene via lyd- og videoopptak, samt fått underskrevet samtykkeerklæring fra foreldrene til elevene. I delkapitlet over har jeg gjort rede for etiske utfordringer og vurderinger.

Det er ikke alltid at forskningens informanter kan verifisere forskerens analyser (Bryman, 2016, s. 385). Elevene som har deltatt som mine intervjuobjekt har fått muntlig beskjed om at de kan se hva jeg skriver i ettertid. Dette er det foreløpig ingen som har etterspurt. Jeg tror ikke de vil ønske å lese alt, jeg tror heller ikke de vil ha forutsetningene for å forstå analysen uten det teoretiske rammeverket.

3.6.2 Overførbarhet (Transferability)

Ettersom kvalitative studier går i dybden, ikke bredden, kan funnene sjelden generaliseres eller gjenskapes. Istedenfor blir kvalitative forskere oppfordret til å gi detaljerte beskrivelser av den sosiale settingen som observeres. Andre som leser forskningen vil da kunne selv avgjøre om forskningen muligens kan overføres til andre miljøer (Bryman, 2016, s. 384, 697).

Kravet til overførbarhet føler jeg jeg har imøtekommet. Jeg har gitt detaljerte beskrivelser av hvordan jeg har tenkt å introdusere undervisningsoppleggene mine og hvilke oppgaver elevene vil få å arbeide med. I analysene mine redegjør jeg for hva elevene sier og tar også med noe av kroppsspråket som kan være med på å tolke hva som menes. Noen steder beskriver jeg hva som skjer samtidig som elevene snakker, eller mellom utsagnene deres.

Det vil ikke være mulig, eller ønskelig, å gjenskape nøyaktig de samme oppgavebaserte intervjuene. Men med detaljene og beskrivelsene som kommer fram i denne oppgaven mener jeg det vil være mulig å gjennomføre tilsvarende opplegg og etterprøve mine funn.

3.6.3 Pålitelighet (Dependability)

For å sikre forskningens pålitelighet foreslås det å begrunne og ta vare på alle avgjørelser underveis i forskningsperioden (Bryman, 2016, s. 384).

Denne oppgaven er grundig strukturert og kronologisk bygd opp. I innledningen beskrives bakgrunnen og begrunnelsen for det valgte forskningsspørsmålet. Jeg har lett fram og presentert aktuell litteratur i kapittel 2 som forskningen kan bygge på. I kapittel 4.1 forklarer jeg hvordan undervisningsoppleggene er planlagt utført, i kapittel 3.3 hvordan jeg foretok utvalget av intervjuobjekter og i kapittel 3.4 hva jeg ser etter i analysen. Det kommer tydelig fram i denne oppgaven hva jeg har sett etter, hvorfor og hvordan. Undervisningsoppleggene og transkripsjon med tidsstempel fra intervjuene er vedlagt. Alt dette er tilgjengelig og gjort rede for.

3.6.4 Bekreftbarhet (Confirmability)

Det anerkjennes at komplett objektivitet er umulig, men det forventes at forskeren har oppført seg i god tro. Forskeren skal ikke velvitende tillate at personlige verdier eller teoretiske overbevisninger påvirker forskningen og dens funn (Bryman, 2016, s. 386).

Jeg er en lærer, som har interesse i og noe erfaring med programmering, som forsker på hvordan programmering kan integreres i matematikkundervisningen. Mitt engasjement tror jeg kommer fram i innledningen. Rollen til en forsker er ikke å undervise, men i denne oppgaven er jo essensen om programmering kan bidra til mer dybdelæring. Derfor mener jeg at min innblanding i undervisningen som medlærer og under intervjuene er begrunnet. I et virkelig klasserom vil elevene ha en lærer til stede som leder undervisningen. Mine dytt og spørsmål til elevene under intervjuene vil ikke være for å bevisst endre utfallet av forskningen, men heller for å se hvilke muligheter programmering kan åpne opp for. Jeg inkluderer i mine analyser hvor jeg deltar i elevenes oppgaveløsing og legger ikke skjul på min egen rolle.

I dette kapitlet har jeg beskrevet og begrunnet hvordan jeg går fram for å besvare forskningsspørsmålet mitt. Jeg har redegjort for mine valg og fremgangsmåter og beskrevet hvordan jeg skal bearbeide datamaterialet jeg samler inn. Jeg har avsluttet dette kapitlet med å skrive om etiske utfordringer og oppgavens troverdighet.

I neste kapittel skriver jeg om mine resultater og analyser. Jeg starter med en beskrivelse av undervisningsoppleggene jeg har utviklet og beskriver hvordan introduksjonen til timen gjennomføres i hel klasse. I de neste delene analyserer jeg de oppgavebaserte intervjuene først med hensyn på den algoritmiske tenkeren og deretter i lys av kjerneelementene. Så vurderer jeg om oppleggene mine har fungert og avslutter med observasjoner gjort under intervjuene i forhold til holdninger som ikke opprinnelig var en del av det teoretiske rammeverket. Samlet vil jeg forhåpentligvis ha grunnlag for å diskutere problemstillingen min.

4 Resultater og analyser

Dette kapitlet handler om mine resultater og analyser. Kapitlet vil jeg dele inn i fem deler. I første del vil jeg presentere de tre undervisningsoppleggene jeg har utviklet til arbeidet mitt med denne oppgaven. Jeg viser til hvordan jeg har tenkt med de fire fordelene (se kapittel 2.4 Utforming av oppgaver) knyttet til programmeringsoppgaver i matematikkundervisningen og analyserer videre målene med opplegget. Jeg gir en detaljert beskrivelse av hvordan jeg har tenkt å introdusere undervisningstimene.

I andre del av dette kapitlet analyserer jeg elevenes arbeid. Her går jeg kronologisk til verks med å beskrive elevenes fremgangsmåter og prosesser i arbeidet med oppgavene og ser etter kjennetegn på algoritmisk tenking (Tabell 4, s. 24). Hvis programmering virkelig er hovedaktiviteten i undervisningsoppleggene jeg har utviklet, vil det bety at elevene arbeider som programmere; altså at de er algoritmiske tenkere og lager algoritmer. Arbeidsmåtene og nøkkelbegrepene knyttet til den algoritmiske tenkeren vil jeg utheve i *kursiv*.

I tredje del vil jeg ta for meg kjerneelementene i matematikk. Her vil jeg arbeide tematisk og ta for meg ett og ett kjerneelement. Elevenes arbeid som jeg har gjennomgått i delkapittel 4.2 Algoritmisk tenking vil analyseres med hensyn på operasjonaliseringen av kjerneelementene (Tabell 5, s. 24). Dersom programmeringskompetanse og matematiskkompetanse kan utvikles parallelt, må jeg kunne vise til at elevene både arbeider som programmere og med kjerneelementene i matematikk samtidig.

I den fjerde delen vurderer jeg om jeg har lykkes med undervisningsoppleggene mine. Jeg tar utgangspunkt i de fire fordelene jeg har planlagt med og kommenterer om de fungerte slik jeg hadde sett for meg.

I siste del presenterer jeg funn i forhold til holdninger som ikke var en del av det teoretiske rammeverket, men som jeg mener bør tas med som en del av resultatene av forskningen min.

4.1 Utvikling av undervisningsopplegg

I arbeidet med å utvikle undervisningsopplegg ønsker jeg å tenke med de fire fordelene (se kapittel 2.4 Utforming av oppgaver) som programmering kan gi: 1) LIST, 2) Konseptuell overraskelse, 3) Vide vegger og 4) Medvirkning. I de neste delkapitlene beskriver jeg hvordan de ulike timene er planlagt til å være i tråd med fordelene nevnt over, hvordan timene blir introdusert og hvilke oppgaver elevene skal arbeide med. Jeg velger å være rimelig detaljert i mine beskrivelser av opplegg slik at konteksten elevene arbeider i er kjent før analysene av elevarbeidet. Valgene jeg tar i utformingen av oppleggene vil påvirke elevene som skal arbeide med dem. En annen grunn til å være detaljert er å nå et av mine mål med denne masteroppgaven: å hjelpe andre lærere med å tilrettelegge for programmering i matematikk.

4.1.1 Scratch – Mangekanter og vinkler

Det første undervisningsopplegget skal være i tilknytning til geometri med programmeringsspråket Scratch. Jeg har selv brukt Scratch en del og ser for meg et opplegg hvor elevene må bruke presise matematiske instrukser for å tegne ulike

mangekanter. De generelle egenskapene til regulære mangekanter kan oppdages, utforskes og abstraheres.

1) LIST

Inngangsterskelen blir lav ved at jeg introduserer programmeringsspråket og gir konkrete eksempler som elevene kan arbeide ut i fra. I Scratch er det ulike kategorier av blokker som har egne fargekoder, dette forenkler også brukeropplevelsen. Ved hjelp av bevegelsesblokker kan vi styre bevegelsene til en katt. Denne katten kan vi få til å «holde» en penn slik at den tegner bevegelsene sine. Vi kan få katten til å gå fremover, snu et visst antall grader og deretter fortsette fremover for å tegne en vinkel. Den store takhøyden i opplegget realiseres ved at det er store muligheter for å oppdage mønster underveis som kan generaliseres og abstraheres.

2) Konseptuell overraskelse

Den konseptuelle overraskelsen jeg planlegger for er at antall hjørner i en regulær mangekant multiplisert med vinkelen katten må snu alltid blir 360°. Dette kan en også se visuelt ved at katten totalt snur seg en hel runde. For eksempel hvis eleven skal bruke katten til å tegne en regulær trekant, vil den måtte snu 120° og 120° multiplisert med 3 blir 360°.

3) Vide vegger

De vide veggene kommer av at Scratch i seg selv er nettbasert og dermed lett kan deles med andre. Elevene arbeider sammen og deler sine opplevelser både med hverandre og med meg som intervjuer. Hvis opplegget gjennomføres som planlagt ser jeg for meg at elevene gjerne vil vise fram arbeidet sitt hjemme.

4) Medvirkning

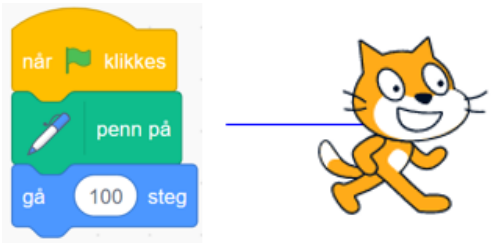
Etter introduksjonen er det hensikten min å overlevere kontrollen til elevene selv. Det er de som skal styre progresjonen sin og de skal selv få velge rekkefølge på oppgavene de får. I Scratch er det mange muligheter og elevene må selv diskutere seg fram til mulige løsninger.

Introduksjon jeg tar felles for hele klassen må følge en tydelig progresjon slik at jeg er sikker på at elevene tilegner seg de nødvendige forkunnskapene for oppgavene.

Steg 1 - Oppsett: Elevene må deles i par og samarbeide om én maskin. De går inn på nettsiden https://scratch.mit.edu/. Derfra klikker vi oss inn på programmering og jeg viser hvordan vi kan legge til Penn-blokkene som er tilleggsfunksjoner. Når alt det tekniske er ordnet, kan vi starte programmeringen.

Steg 2 – Tegne linjestykke: Jeg viser de ulike delene av brukergrensesnittet. Blokkene er sortert i kategorier med ulike fargekoder. Det grønne flagget oppe til høyre kan vi trykke på for å kjøre et program. Den første blokken i et program kan derfor være «Når grønt flagg klikkes», denne blokken drar vi ut til skript-området vårt hvor vi pusler sammen algoritmene våre. Hvis vi trykker på det grønne flagget nå, skjer det ingenting. Hvorfor ikke? Jeg ønsker å legge til rette for en dialogpreget introduksjon, hvor elevene er delaktige i hva som blir gjort på skjermen. De må selv komme med forslagene. Videre

finner vi sammen ut av hvilke blokker vi kan bruke for å tegne et linjestykke (Figur 2).

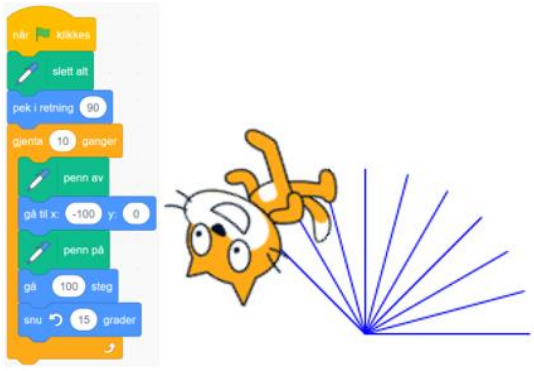


Figur 2: Et linjestykke i Scratch

Steg 3 – Det samme hver gang: Hvis vi trykker på det grønne flagget flere ganger, vil katten bare bevege seg lenger og lenger bortover. Når vi lager et program er det ofte ønskelig at den skal gjøre det samme hver gang den kjøres, i vårt tilfelle skal det bare tegnes ett linjestykke, ikke mange etter hverandre. For å få dette til kan elevene lese på noen av blokkene, de vil helt sikkert finne «slett alt» under kategorien «Penn». Sammen kan vi prøve og feile med hvor blokken skal plasseres i koden vår. I programmering er feil en helt naturlig del av prosessen. Resultatet av det vi har gjort får vi se med en gang, og kan dermed raskt avgjøre om vi har lykkes eller om vi må prøve igjen. Før jeg sier meg fornøyd, bør vi få katten til å starte på det samme stedet hver gang. Da ender vi opp med kode som ligner på Figur 3. Da får vi det samme resultatet hver gang det grønne flagget klikkes og programmet kjøres.

Figur 3: Det samme hver gang

Steg 4 – Løkke: Det neste elevene bør bli kjent med er et viktig programmeringsprinsipp; løkker. Vi kan få katten vår til å gjenta noe flere ganger. Jeg kan plassere koden vi alt har skrevet i en løkke som gjentas 10 ganger, men elevene vil ikke se noen endring av resultatet med det første. Elevene vil sikkert foreslå å fjerne «slett alt» blokken fra løkken, den kan plasseres utenfor, men fortsatt vil ikke resultatet endres. Sammen finner vi ut av at katten tegner det samme linjestykket flere ganger oppå hverandre. Jeg legger da inn en «snu» blokk, slik at det blir en endring, og en «pek i retning» blokk tidlig i koden slik at vi alltid har det samme utgangspunktet (Figur 4).

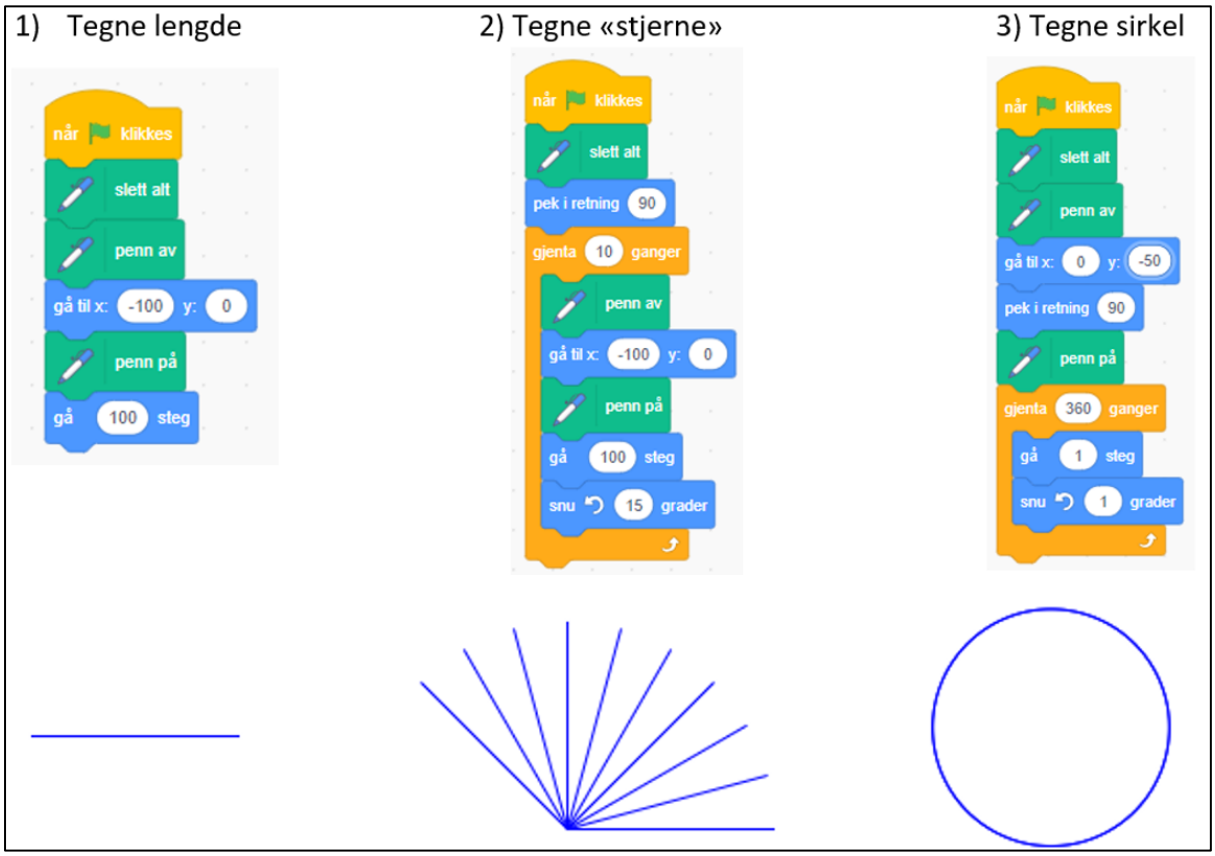


Figur 4: Løkke

Steg 5 – «Stjerne»: Et passende spørsmål her kan være hvor mange ganger må løkken gjentas for at katten skal komme helt rundt og fullføre en «stjerne»? Her bruker vi sannsynligvis prøv og feil strategien, elevene kan komme med forslag, så ser vi hva som skjer. Når «stjernen» er komplett kan jeg spørre hvorfor akkurat 24 gjentakelser? Her er det mulig at noen kobler at 15° 24 ganger blir 360°.

Steg 6 – Sirkel: Til slutt i introduksjonen utfordrer jeg klassen til å fortelle meg hvordan vi kan tegne en sirkel. Noe av diskusjonen vil da være at vi ønsker å tegne videre etter hver gang katten snur, så «gå til» blokken må ut av løkken. Deretter forventer jeg at noen vil oppgi at i en sirkel er det 360 grader. Denne informasjonen ønsker jeg at alle skal få repetert, for den er viktig senere i opplegget. En mulig løsning for å tegne en sirkel er da å snu 1° 360 ganger. Siden vi er mest interessert i linjestykkene og ikke katten, kan jeg vise elevene hvor vi kan trykke for å gjøre katten usynlig.

Jeg ønsker ikke å bruke for lang tid på introduksjonen, men det er en del elevene må bli kjent med. Med en så tydelig plan for progresjonen min vil jeg tro jeg klarer det på et kvarter. Elevene vil få ut en kopi av eksemplene vi har gjennomgått sammen, slik at de har noe å se til om det trengs for å repetere (Figur 5).



Figur 5: Introduksjonen

Etter introduksjonen får elevene i par utdelt oppgaveark og får beskjed om at de skal samarbeide på én maskin. Øverst på oppgavearket finner de eksemplene som vi nettopp har gjennomgått. Oppgave 1 er:

1 Løs oppgavene, bestem selv rekkefølge

a) Tegn en trekant
b) Tegn en firkant
c) Tegn en femkant
d) Tegn en sekskant
e) Hvilke andre former klarer du å tegne?

Figur 6: Oppgave 1

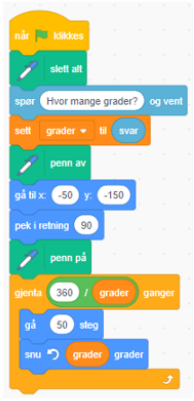
Jeg ønsker at elevene skal trives med oppgavene og få lov til å være kreative. Derfor kan de selv bestemme rekkefølge, og oppgave e) er svært åpen. Dersom de ikke klarer, eller ikke vil tegne mangekanter kan de tegne andre former uten at det de gjør er «feil». Jeg er sikker på at det uansett vil dukke opp matematiske diskusjoner om vinkler og lengder.

Det står ikke at elevene skal tegne regulære mangekanter, men ettersom jeg har introdusert dem for løkker, regner jeg med at de prøver å tegne mangekanter som er regulære. Grunnen til dette er at hvis «snu»-blokken ligger i løkken vil katten snu det samme antall grader hver gang det lages en ny vinkel. For å tegne mangekanter som ikke er regulære må elevene legge til flere «snu»-blokker med forskjellige antall grader. Hadde jeg ikke gjennomgått løkker i introduksjonen, kunne dette fort ha vært aktuelt, men siden jeg allerede har vist dem eksempelkode med løkker vil dette være enklere.

Når elevene forsøker å tegne en trekant, mistenker jeg at de vil forsøke å snu katten 60°, da jeg tror de vet at vinkelen i en likesidet trekant er nettopp dette. Men å snu 60° vil ikke resultere i en trekant. Den innvendige vinkelen er 60°, men antall grader katten må snu er ikke det samme som den innvendige vinkelen. Det er nabovinkelen. Jeg er spent på hvordan elevene løser dette.

Videre håper jeg at elevene oppdager noen mønster etter hvert som de får tegnet flere av mangekantene. Hvis de gjør det kan de få utdelt oppgave 2) og 3). Jeg velger å ikke dele ut alle oppgavene samtidig, for da vil elevene kanskje tenke at målet er å bli ferdig med alle oppgavene. Målet er å bruke programmering til å tilegne seg matematisk kunnskap. Dersom elevene oppdager noen mønster mellom antall hjørner og antall grader katten må snu, er de kanskje klare for å generalisere og abstrahere. Dersom de ikke oppdager mønstrene ønsker jeg ikke at de skal bevege seg videre til de neste oppgavene for tidlig. Det forblir opptil læreren å avgjøre når elevene er klare.

2 Hva gjør dette programmet?



3 Klarer du å lage et program som ber brukeren oppgi antall hjørner, og deretter tegner en mangekant med det gitte antall hjørner?

F.eks:
- Programmet spør: «Hvor mange hjørner?»
- Brukeren svarer: «7»
- *Programmet tegner en 7-kant.*

Figur 7: Oppgave 2 og 3

Poenget med oppgave 2) er å introdusere elevene for hvordan vi i Scratch kan be brukeren om «input» og lagre svaret i en variabel. Deretter kan variabelen brukes i kalkulasjoner. Fargekodene bør gjøre det overkommelig for elevene å kopiere koden på egen skjerm, men her er jeg uansett forberedt på å måtte gi noe teknisk veiledning.

Hvis elevene klarer å komme seg til oppgave 3) er jeg veldig fornøyd. Her må elevene koble sammen de nye programmeringsblokkene fra oppgave 2) med de matematiske mønstrene de forhåpentligvis har oppdaget i arbeidet med oppgave 1). Klarer de dette har jeg sannsynligvis fått mye data som vil peke mot at programmering har gitt elevene dypere innsikt i mangekantenes geometri.

4.1.2 Scratch – Lineære funksjoner

Det andre undervisningsopplegget skal være en introduksjon til lineære funksjoner, etter ønske fra klassens lærer, og igjen velger jeg å bruke programmeringsspråket Scratch, men i tillegg også GeoGebra (som graftegner) som jeg vet elevene har brukt før. Programmering er vel-egnet til å skape en forståelse for bruken av variabler og tilbyr en helt annen inngang til algebra enn å trekke sammen algebraiske uttrykk, slik en ofte møter det i lærebøker. I dette opplegget vil elevene lære å opprette og bruke egne variabler samt utforske lineære sammenhenger.

1) LIST

Scratch er i seg selv et lavterskel programmeringsspråk med stor takhøyde. Dette er allerede beskrevet tidligere, så jeg går ikke videre inn på det her. Jeg har planlagt å bruke en «funksjonsmaskin» som min inngang til lineære funksjoner. En funksjonsmaskin er en måte å illustrere at en verdi går inn, deretter skjer det en beregning, og så kommer en verdi ut. Verdien som går inn kan elevene bestemme, verdien som kommer ut er resultatet av en beregning som utføres likt fra gang til gang. Etter at elevene har sett et par eksempler blir oppgaven deres å se mønsteret og gjette hva utverdien blir etter hvert som vi bestemmer nye innverdier. Elevene skal finne funksjonsuttrykket. Denne introduksjonen har lav inngangsterskel fordi alle elever kan være med på å bestemme hvilke tall som skal inn i maskinen, og gjette hvilken verdi som kommer ut. Videre vil jeg lage to programmer sammen med hele klassen fra bunnen av, slik at alle har mulighet til å tilegne seg nok kunnskaper til å gå videre i opplegget. Elevenes hovedoppgave blir å lage en egen funksjonsmaskin i Scratch basert på en mal som de får. Deretter skal maskinens inn- og utverdier plottes i et koordinatsystem ved hjelp av GeoGebra. Opplegget er ikke særlig rikt siden det er tydelige føringer for hvordan oppgavene skal løses. Den store takhøyden kommer først i forlengelsen av denne timen og diskusjonene som oppstår etter at flere av funksjonene blir plottet inn i et koordinatsystem. Da kan det stilles spørsmål som: Hvorfor stiger noen grafer mer enn andre? Hvorfor er noen parallelle? En time blir knapt med tid, så jeg har kuttet og tatt noen snarveier for å sikre at alle, uansett programmeringsbakgrunn, kan få det til. Denne timen kan danne grunnlaget for videre utforsking.

2) Konseptuell overraskelse

Alle funksjonsmaskinens inn- og utverdier beskriver koordinater til punkt som kan plottes i et koordinatsystem. Overraskelsen jeg planlegger for er at punktene som plottes i GeoGebra danner en rett linje i koordinatsystemet. Vil det alltid være slik? Det

er mulig at noen lager en funksjon som ikke er lineær, men slik programmet mitt er laget er det usannsynlig. Det kan derimot være en fin utfordring på sikt, er det mulig å lage en funksjonsmaskin som ikke gir lineære funksjoner?

3) Vide vegger

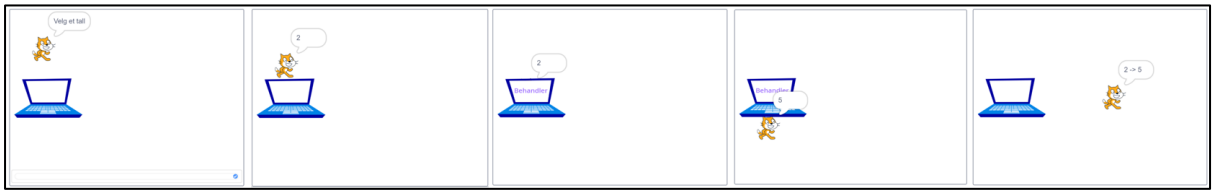
Her er det igjen i stor grad valget av Scratch som programmeringsspråk som sørger for de vide veggene. Programmet er nettbasert og gratis, alle kan lage seg en bruker hvor de kan lagre koden de skriver og dele den med andre. En del av opplegget er også at elevene skal dele sine funksjonsmaskiner med resten av klassen og dermed åpne opp for samhandling mellom elevene. De kan dele sine oppdagelser og frustrasjoner og lære av hverandre.

4) Medvirkning

I dette opplegget blir elevene først bedt om å delta i en felles funksjonsmaskin utprøving, eleven velger selv hvilke verdier som skal inn. Deretter blir de bedt om å løse «prosedyre-oppgaver», der er det ikke elevene som styrer fremgangen. Etter dette får de igjen anledning til å bestemme mer og ta mer kontroll ettersom de forhåpentligvis forstår hvordan koden er bygd opp. De kan gjøre endringer og se resultatet med en gang og i en tenkt neste time vil de kunne ta enda mer kontroll og utforske sammenhenger mellom funksjonsuttrykk og graf.

Felles introduksjon for hele klassen vil begynne uten at elevene er på sine egne datamaskiner. Igjen legger jeg opp til dialogisk undervisning hvor elevene hjelper meg med å gjennomføre undervisningen.

Steg 1 – Funksjonsmaskin: Jeg vil presentere en funksjonsmaskin som jeg har laget i Scratch (https://scratch.mit.edu/projects/492452630). Elevene vil foreslå en verdi, katten sier verdien og beveger seg gjennom en maskin som behandler verdien og gir katten en ny verdi å si når den kommer ut av maskinen (Figur 8). Til slutt sier katten hvilken verdi som gikk inn, og hvilken som kom ut. Dette skriver lærer på tavlen i klasserommet, slik at det blir tatt vare på. Deretter kan en starte programmet på nytt og velge en ny verdi.



Figur 8: Funksjonsmaskin i Scratch

Steg 2 – Ta fram egne datamaskiner: Når elevene har gjettet funksjonsuttrykket, får de beskjed om at målet i denne timen er at de skal lage sine egne funksjonsmaskiner i Scratch og de skal lære om variabler. Men først må de bli kjent med noen av blokkene som de skal bruke. De deles i par og arbeider sammen på én maskin. De går inn på https://scratch.mit.edu/.

Steg 3 – Få katten til å si noe: Elevene har alt sett at programmet kjøres når en trykker på det grønne flagget, så vi starter med den blokken som heter «når grønt flagg klikkes». Deretter forklarer jeg at blokkene tilhører forskjellige kategorier som har sine egne

fargekoder. Hvordan kan vi få katten til å si noe? Ved å trykke litt på kategoriene finner elevene fort «si»-blokken under kategorien «Utseende».

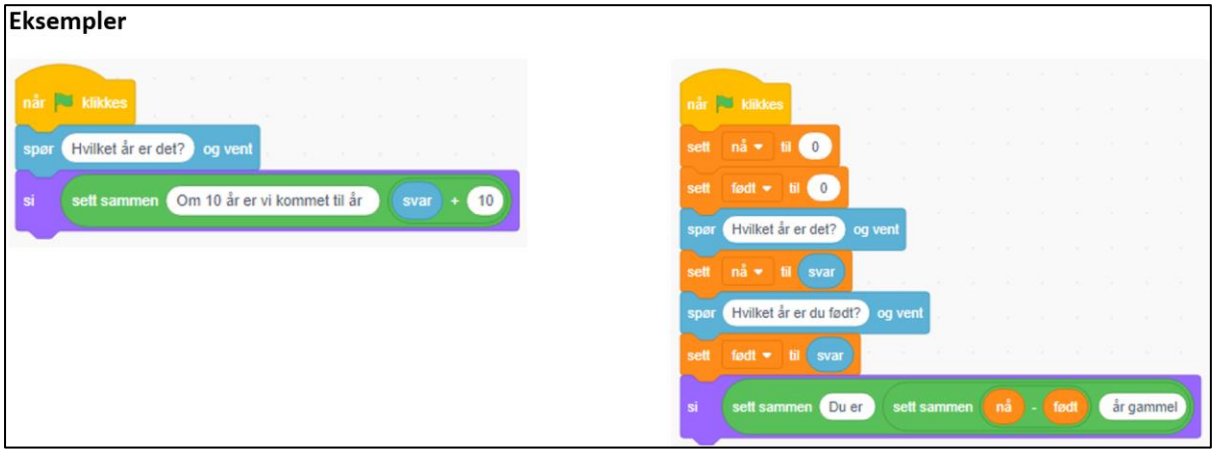
Steg 4 – Få katten til å si noe som brukeren bestemmer: Slik funksjonsmaskinen fungerte, kunne elevene selv velge verdien katten skulle starte med. Hvordan kan jeg få katten til å si noe som brukeren av programmet bestemmer? Vi må stille et spørsmål til brukeren, f.eks. «Hvilket år er det?». Her viser jeg elevene hvor vi kan finne «spør og vent»-blokken i kategorien «Sansing» og hvordan vi kan bruke «svar»-blokken til å få katten til å si det brukeren skriver inn (Figur 9).

Figur 9: Spør og vent

Steg 5 – Utføre en beregning med «svar»: «svar»-blokken er en variabel. Her oppstår en fin mulighet til å samtale om hvorfor det kalles en variabel. Videre kan jeg spørre om programmet mitt kan beregne hvilket årstall det vil være om ti år basert på årstallet brukeren av programmet skriver inn. Elevene vil nok kjapt muntlig kunne komme med løsninger og jeg kan vise hvilke blokker som kan brukes for å få det til i Scratch. Elevene utfører den algoritmiske tenkingen, mens jeg skriver selve algoritmen.

Steg 6 – Utføre beregning med to «svar» fra brukeren: Sammen prøver vi å lage et eksempel hvor brukeren av programmet skal oppgi to verdier som programmet skal bruke i sine beregninger. Et problem som da oppstår er at det kun er én «svar»-blokk, slik at når vi stiller det andre spørsmålet, vil svaret der overskrive det første svaret. Hva gjør vi nå? Gjennom dialog kommer vi nok fram til at det er mulig å opprette en ny variabel for å ta vare på den første verdien, eller to nye variabler for å ta vare på begge verdiene. Deretter kan de nye variablene brukes i beregningene vi ønsker å utføre.

Figur 10 viser eksempler på kode vi kan skrive for å illustrere steg 5 og 6 av introduksjonen og som elevene får utdelt når de skal arbeide med egne oppgaver.



Figur 10: Beregninger med variabler

Denne introduksjonen ser jeg vil ta noe tid. Undervisningstimen er en klokketime, jeg håper det vil holde til å fullføre hele opplegget.

Etter introduksjonen får elevene i par utdelt oppgaveark og får beskjed om at de skal samarbeide på én maskin. Øverst på oppgavearket finner de eksemplene som vi nettopp har gjennomgått. Oppgavene er:

a) Lag et program som spør brukeren hvor gammel hun er og som deretter svarer hvor gammel brukeren vil være om 10 år.

b) Lag et program som spør brukeren om fornavnet sitt, deretter om etternavnet sitt og som deretter svarer det fulle navnet til brukeren.

c) Følg linken: https://scratch.mit.edu/projects/341400228/editor/
Fyll inn klosser i de tomme boksene som vil gi ønsket resultat.

d) Gjør din egen endring på «funksjonsmaskinen» slik at maskinen regner ut andre verdier enn originalen. Eksempel:

Figur 11: Elevenes oppgaver

Målet med oppgave a) og b) er at elevene skal bruke blokkene fra eksemplene, slik at de kommer i gang med Scratch. Det vil ta for mye tid og være for krevende å be elever uten tidligere erfaringer med programmering og Scratch om å lage sine helt egne funksjonsmaskiner fra bunnen (selv om dette kunne vært en fin utfordring for mer viderekomne programmere), derfor gir jeg dem min versjon i oppgave c) (https://scratch.mit.edu/projects/341400228/editor/). For å sikre at de forstår de ulike delene av koden har jeg fjernet noen blokker slik at de selv må finne ut av hvordan de skal få programmet til å fungere som i oppstarten av timen. Til slutt i oppgave d) må de finne ut av hvor de må gjøre endringer for å lage sin egen versjon av funksjonsmaskinen.

De som blir ferdige først kan utfordre hverandre til å gjette funksjonsuttrykket til stadig nye funksjonsmaskiner. Etter hvert som flere blir ferdige er planen at læreren tar en felles oppsummering. Elevene kan vise fram funksjonsmaskinene. Innverdier og

utverdier skrives strukturert opp på tavlen sammen med det første eksempelet fra oppstarten.

Til slutt er planen at læreren tar fram GeoGebra og plotter inn verdiene fra de ulike funksjonsmaskinene som koordinater i et koordinatsystem (dette må muligens bli i timen etter). Hva legger elevene merke til? På grunn av hvordan jeg har laget malen til funksjonsmaskinen er sannsynligheten stor for at alle funksjonsmaskinene gir lineære funksjoner; alle punktene til en maskin danner en rett linje. Dette kan danne grunnlaget for videre diskusjoner.

4.1.3 p5.js – Sannsynlighet

Nå har jeg opprettet to opplegg med Scratch som programmeringsspråk. Etter hvert som elevene tilegner seg mer programmeringskompetanse vil det være ønskelig å bevege seg videre til et tekstbasert programmeringsspråk. Jeg vil at mitt siste opplegg skal ta utgangspunkt i tekstbasert programmering og skal lage et opplegg som forhåpentligvis er overkommelig selv for en elev som ikke har vært borti programmering før. Et av de nye kompetansemålene i læreplanen er at elevene skal kunne: «simulere utfall i tilfeldige forsøk og berekne sannsynet for at noko skal inntreffe, ved å bruke programmering» (Utdanningsdirektoratet, 2020b). Her lager jeg et opplegg med utgangspunkt i det.

1) LIST

Utgangspunktet for denne timen er et «hesteløp». Vi skal starte med å simulere kast av en terning og lage seks virtuelle «hester» som beveger seg et skritt fremover hvis tallet sitt dukker opp. Deretter skal elevene selv simulere kast med to terninger hvor summen av de to terningene avgjør hvilken «hest» som skal flytte et steg framover. Hesteløpet skal simuleres i p5.js. Dette opplegget har lav inngangsterskel ettersom alle elevene kan forstå oppgavens kontekst. Et hesteløp kan alle forstå hva er og dermed har de tydelige bilder av hva det er som skal skje. Elevene får anledning til å stille spørsmål, og kan se resultatet av programmet visuelt. Tekstbasert programmering kan være utfordrende, derfor får de et eksempel tydelig gjennomgått som de kan videreutvikle etterpå. Det er stor takhøyde fordi elevene må gjenkjenne mønster og reflektere over hvorfor de oppstår. Elevene er sannsynligvis ikke kjent med så mye programmering fra før og får muligheten til å skape og forstå noe som ved første øyekast kan se veldig komplisert ut. Elevene må benytte seg av viktige programmeringsprinsipp i sitt arbeid som løkker, variabler og vilkår. Opplegget gir gode forutsetninger for å diskutere forskjellen på en statistisk sannsynlighetsmodell og en uniform sannsynlighetsmodell. Videre kan programmet brukes til å diskutere den store talls lov og/eller elevene kan utfordres til å gjøre hesteløpet mer rettferdig ved å f.eks. tillate at noen av hestene får gå flere steg fram hvis terningene viser deres sum. Mulighetene er mange.

2) Konseptuell overraskelse

At sannsynligheten er den samme for at alle «hestene» kan vinne når det kun kastes én terning, forventer jeg er kjent stoff. Men at det blir annerledes ved kast av to terninger tror jeg kan bli en ny erfaring for elevene. Når formen til hestene danner en slags trekant, hvor 7-er «hesten» stikker av med seieren, håper jeg elevene får en konseptuell overraskelse (hvis de ikke har vært borti problemstillingen tidligere, vel og merke). Jeg håper også at elevene er i stand til å forklare hvorfor denne formen oppstår. Elevene må

sette seg grundig inn i problemet, og virkelig forstå det for å kunne lage programmet. Derfor tror jeg også de vil kunne forklare denne overraskelsen.

3) Vide vegger

p5.js er i likhet med Scratch nettbasert og tilgjengelig for alle. Det er enkelt å opprette en konto slik at programmer kan lagres og deles med andre. Elevene kan kopiere en lenke og sende til hvem som helst. Alle parene i klasserommet vil nok være i en felles problemløsingsfase og kan dele erfaringer og forslag med hverandre. Jeg ser for meg at dette er et program å være stolt av og som en sikkert vil vise fram hjemme.

4) Medvirkning

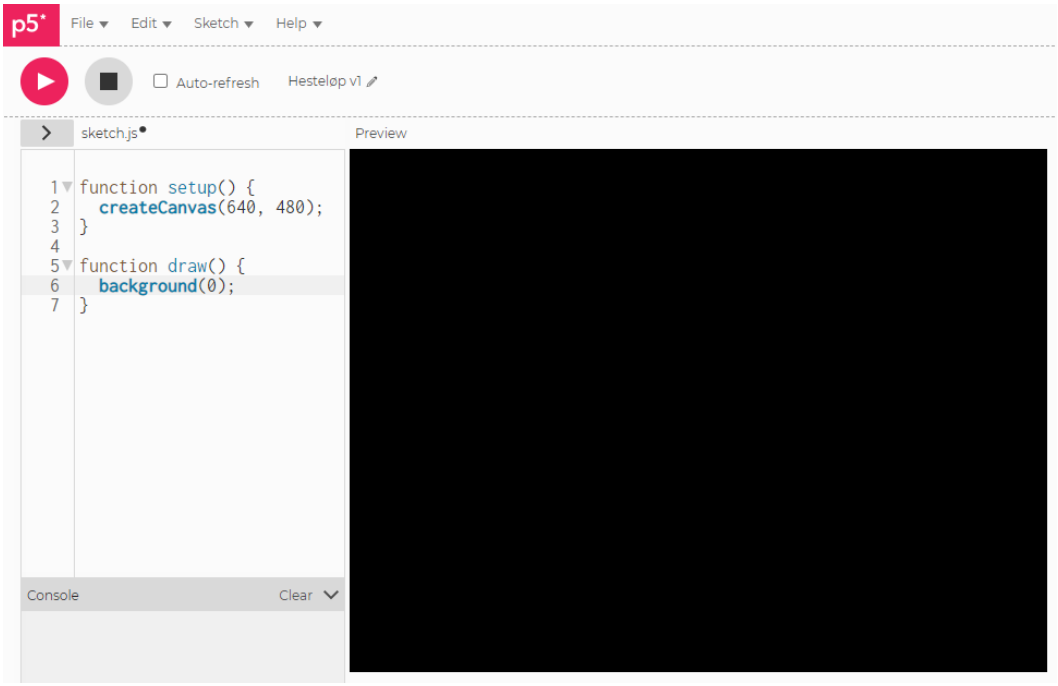
I starten er det læreren som styrer mye av dette opplegget for å sikre at alle blir med på reisen. Når elevene blir kjent med programmeringsspråket og -prinsipper, slippes de løs. Da er det elevene som må styre og tenke selv for å lage et program som passer til bestillingen. De gjør nok mange feil på veien, men kan oppdage det raskt ettersom programmet gir umiddelbar respons. Nederst i editoren til p5.js er det et konsollvindu som forteller om oppdagede feil i koden og mulige forklaringer til hvorfor feilen har oppstått. Jeg ser for meg at dette kan bli en engasjerende aktivitet.

Den første delen av timen kommer til å bli lærerstyrt. Elevene får muligheten til å komme med innspill og stille- og svare på spørsmål. Introduksjonen må ha en strukturert og tydelig progresjon. Elevene skal tilegne seg mye nytt på kort tid, men forhåpentligvis vil den kjente situasjonen gjøre det overkommelig for elevene.

Steg 1 – Oppsett: Elevene setter seg i par med en datamaskin, men får ikke bruke den enda. Lærer går inn på https://editor.p5js.org/ via egen maskin som er koblet opp mot en projektor slik at alle kan se. Her blir de møtt med tre hovedvinduer. Øverst til venstre er det allerede skrevet inn noe kode, her kommer vi til å fylle på med våre egne koder etterhvert. Under dette vinduet er det et konsollvindu som kan fortelle oss om feilene i koden vår. Vinduet til høyre er forhåndsvisningen (visningsvinduet) som viser hva programmet vårt gjør. For å kjøre programmet må vi trykke på «play» knappen over vinduene til venstre.

Steg 2 – Hesteløp: Lærer beskriver hvordan hesteløpet med en terning fungerer og ber elevene velge en hest som de vil satse på. En ferdigprogrammert versjon av hesteløpet med én terning kjøres og en vinner kåres. Vi skal nå starte fra bunnen av og lage dette programmet.

Steg 3 – Svart bakgrunn: Kodene som alt står gir oss en ypperlig anledning til å utforske; Hva gjør disse kodene? Vi kan kjøre programmet vårt, endre på verdiene som står, kjøre programmet på nytt og legge merke til endringene. For å få svart bakgrunn må verdien inni «background» settes til 0 (Figur 12).



Figur 12: Svart bakgrunn p5.js

Steg 4 – Lage en terning: Lærer må skrive nokså fort, men samtidig bruke tid på å forklare. Det første vi gjør er å opprette en variabel som vi kaller terning1. Hvilke utfall kan vi få når vi kaster en vanlig terning? Utfallene skriver vi inn i en tabell. Både variabelen og tabellen skal skrives øverst i koden vår (Figur 13).

```
var terning1;
var antallOyne = [1,2,3,4,5,6];
```

Figur 13: Lage en terning

Steg 5 – Heste-variabler: Vi må opprette en variabel for hver hest som angir hvor den skal starte løpet. Hestene starter med posisjon 0 (x-koordinat).

Steg 6 – Kaste terningen: For å kaste terningen må vi tilordne en tilfeldig verdi fra tabellen «antallOyne» til terningen «terning1». Dette høres komplisert ut, men løses ved enkel kode (Figur 14).

```
  //Kast terning1
  terning1 = random(antallOyne);
```

Figur 14: Kast terning1

Steg 7 – Beveg hesten: Nå har vi kastet terningen, men hvilken hest skal flytte på seg? Her må elevene introduseres for *vilkår*, et grunnleggende programmeringsprinsipp; hvis en betingelse er oppfylt, så utføres en kodesnutt. Dersom betingelsen ikke er oppfylt, hopper programmet over kodesnutten. Elevene kan være med på å forklare at hvis den virtuelle terningen viser ett øye, er det hest1 som skal flytte seg ett steg fram (Figur 15). Dette gjentar vi for alle hestene.

```
  //Hvilken hest skal flytte?
  if(terning1 == 1)
    hest1 += 1;
```

Figur 15: Flytt hest1

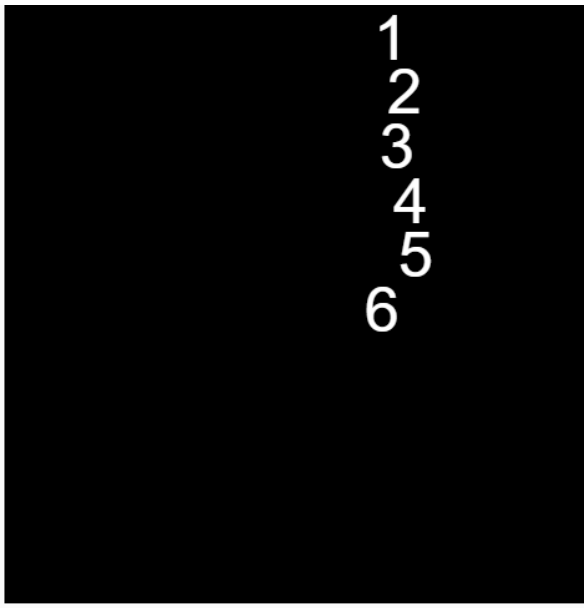
Steg 8 – Se hestene: Foreløpig skjer det ingenting hvis vi kjører programmet vårt. Elevene vet hva de forventer skal skje, men alt er svart. Vi må først vise hestene, da vil vi se at de flytter seg. Først angir vi farge, så oppretter vi en tekst (tall) for hver hest. Hver hest vil vi se som ett hvitt stort tall. x-koordinaten til hest nr. 1 setter vi til variabelen hest1, y-koordinaten har jeg bestemt skal være vinduets maksimale høyde dividert på 11 (på sikt skal vi ha plass til 11 hester). Koden blir da som vist i Figur 16, og vi gjentar for alle hestene, men med økte y-koordinater.

```
  //Vis og flytt hestene
  textSize(50)
  fill(255);

  text(1, hest1, height/11);
```

Figur 16: Se hesten

Steg 9 – Hestene flytter på seg: All koden som tilhører den på forhåndsskrevne funksjonen «draw» gjentas i en *løkke*. «draw» er i grunn en løkke. Det betyr at terningen nå kastes om og om igjen, og hestene vil flytte seg. Programmet er ferdig! Hestene beveger seg mot høyre etter hvert som terningen viser deres verdi (Figur 17).

```
//Oppretter terning
var terning1;
var antallOyne = [1,2,3,4,5,6];

//Oppretter variabler for hestenes plassering
var hest1 = 0;
var hest2 = 0;
var hest3 = 0;
var hest4 = 0;
var hest5 = 0;
var hest6 = 0;

function setup() {
  createCanvas(640, 480);
}

function draw() {
  background(0);

  //Kast terning1
  terning1 = random(antallOyne);

  //Hvilken hest skal flytte?
  if(terning1 == 1)
    hest1 += 1;
  else if(terning1 == 2)
    hest2 += 1;
  else if(terning1 == 3)
    hest3 += 1;
  else if(terning1 == 4)
    hest4 += 1;
  else if(terning1 == 5)
    hest5 += 1;
  else if(terning1 == 6)
    hest6 += 1;

  //Vis og flytt hestene
  textSize(50)
  fill(255);

  text(1, hest1, height/11);
  text(2, hest2, 2 * height/11);
  text(3, hest3, 3 * height/11);
  text(4, hest4, 4 * height/11);
  text(5, hest5, 5 * height/11);
  text(6, hest6, 6 * height/11);
}
```



Figur 17: Hesteløp med en terning

Når vi kommer i mål med dette eksempelet får elevene oppgavearket (8.5 Oppgaveark: p5.js – Sannsynlighet) som de skal arbeide med i par på én maskin. Øverst på oppgavearket står eksempelet vi har laget. Jeg sender dem også lenken til programmet vi nettopp har laget slik at de slipper å skrive av alt.

Elevene får vite at deres oppgave er å lage et program som viser et slikt hesteløp, men hvor det er to terninger som skal kastes og hvor det er summen på antall øyne de to terningene viser som avgjør hvilken hest som skal flytte fram. Oppgavene er en videreutvikling av programmet vi sammen har laget:

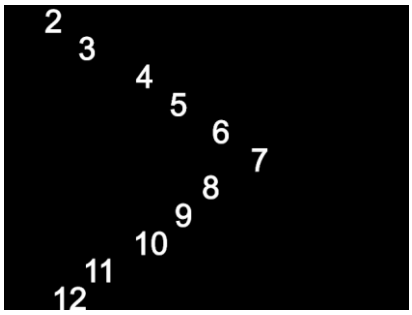
a) Hvis vi kaster to terninger og summerer sammen antall øyne på begge terningene, hvor mange ulike summer kan vi få?
b) Hvor mange «hester» må vi ha i hesteløpet vårt?

c) Lag et program som simulerer hesteløpet med to terninger.

d) Kjør programmet!
Hva legger du merke til?
Hvorfor blir det slik?

Figur 18: p5.js oppgaver

Hvis elevene får til å lage programmet, vil de se at «hest7» har størst sannsynlighet for å vinne og at hestene danner en slags trekant form (Figur 19).



Figur 19: Konseptuell overraskelse

Datamaterialet innhentet med opplegg 1 og 2 har jeg brukt i andre fagartikler i emner jeg har tatt ved UiA i forbindelse med min masterutdanning (Candasamy, 2019a, 2019b). Begge artiklene handler om programmering i matematikkundervisningen, men fokuset til analysene har vært noe annerledes enn i denne oppgaven. I den første artikkelen skrev jeg om *elevers begrepsmessige forståelse av mangekanter gjennom programmering i Scratch*, hvor noe av rammeverket var van Hieles nivåer for geometriforståelse. I den andre artikkelen skrev jeg om elevenes instrumentelle skapelse; hvordan de tilegnet seg kunnskaper om å bruke Scratch til å løse matematiske problem. Jeg vurderte *mulighetene og begrensningene* av Scratch som verktøy slik de fremstod hos elevene.

4.2 Algoritmisk tenking

Her går jeg kronologisk frem og presenterer mine utvalgte elevers arbeid med oppgavene. Noen episoder vil jeg beskrive med mine egne ord med bakgrunn i det jeg ser og hører, andre ganger gjengir jeg utsagn direkte fra transkripsjonen som beskriver progresjonen. Underveis viser jeg utklipp av skjermarbeidet fra både skript- og visningsvinduet til elevene. I det siste opplegget har jeg ikke lyd eller video å støtte meg til, så her vil jeg beskrive hva som har skjedd slik jeg husker det basert på mine egne notater fra timene.

Hensikten med dette delkapitlet er å presentere hva elevene har gjort og knytte dette opp mot algoritmisk tenking (uthevet i *kursiv*).

4.2.1 Scratch – Mangekanter og vinkler

De to jentene som ble tilfeldig valgt til å være mine intervjuobjekter virket lykkelige for å ha blitt trukket ut og gledet seg til å komme i gang. Heretter kaller jeg dem «Kari» og «Trine». Gjennom hele introduksjonen med klassen var de engasjerte og kom med forslag til hvordan blokkene kunne pusles sammen til å få de ønskede resultatene. At det var 360° i en sirkel var godt kjent. De hadde lite erfaring med Scratch fra før.

Etter introduksjonen ble de med meg ut på et grupperom og fikk første del av oppgavearket (8.3 Oppgaveark: Scratch – Mangekanter og vinkler). Jeg ba dem tenke høyt mens de løste oppgavene og gjerne fortelle hva de forventer å se før de kjører programmet sitt.

Den første blokken elevene finner frem er en hendelsesblokk: «når grønt flagg klikkes». Når denne hendelsen inntreffer, det grønne flagget klikkes, starter programmet og utfører koden som elevene skriver/pusler. Elevene går umiddelbart inn i en problemløsingsfase. De deler tankene sine med hverandre og blir enige om hva de skal gjøre; de *samarbeider*.

Trine [00:26] Skal vi endre pennens farge? (smiler/ler)

Kari [00:29] Ja!

[00:33] Ja, vi må ha … Hvilken farge må vi ha?

(Trine endrer fargen til rosa)

[00:46] Ok. Og så … Må den jo gå sånn 100 steg kanskje.

Å *fikle* vil ifølge Tabell 4: Den algoritmiske tenkeren (s. 24) si å eksperimentere og utforske. Det er det jentene gjør når de bytter farge på pennen. De tester ut ulike farger og ender til slutt på rosa. Dette var ikke en del av oppgaven og ikke en blokk jeg hadde vist fram under introduksjonen. Videre bestemmer de seg for hvor mange steg fram katten skal bevege seg, de beslutter at den skal gå 100 steg. Disse blokkene gjør hver for seg noe elevene ønsker. Ved å koble sammen blokkene er elevene i gang med å skrive steg-for-steg instrukser; *algoritmer.*

I oppgave 1a) skulle elevene få programmet til å tegne en trekant. Det står ikke at det skal være en likesidet trekant, men som sagt tidligere var det dette jeg antok at de kom til å prøve seg på ettersom jeg hadde introdusert dem for løkker ved hjelp av «gjenta» blokken.

Kari [00:58] Så må den snu 60 grader, for hvis du skal ha en likesidet trekant så må den jo være 60.

[01:06] [???] Er det ikke bare sånn gjenta …

Trine [01:09] Tre ganger.

Jeg forventet at de kjente til at innvendig vinkel i en likesidet trekant er 60° og at de ville instruere katten til å snu seg 60° for å tegne trekanten.

Kari [01:37] Vi tror at siden den skal gå [???] steg, så flytter den seg 100 piksler. Og så gjør vi den liten (endrer størrelsen på Felix), sånn.

Trine [01:47] Synes den var litt liten.

Kari [01:48] Nei, nei… Nå kan vi se at den er der. Så kommer han til å gå 100 steg og så kommer han til å vri seg 60 grader og så kommer han til å gå 100 steg, og så kommer han til å vri seg 60 grader, så kommer han til å gå 100 steg til.

Trine [01:58] Og så blir det en trekant.

Elevene *dekomponerer*; de bryter trekanten ned i mindre deler; først gå 100 steg, deretter snu 60°. Dette skal gjentas tre ganger. Disse blokkene blir også en del av *algoritmen*. Å bruke l*ogikk* vil si at elevene analyserer og forutser. Her viser de eksempler på nettopp det; de har plukket ut egenskapene til en likesidet trekant (tre like lange sider, i dette tilfellet 100 steg, og tre like store vinkler på 60°) og forutsett at hvis katten gjennomfører akkurat disse stegene så vil resultatet bli en trekant. Elevene arbeider godt sammen, deler tankene sine og fullfører hverandres tankerekker.

Kari [02:00] Ja, men det kan også være at den ikke vrir seg innover, at han bare blir sånn (viser med fingeren at det ikke blir en trekant).

Trine [02:03] Det er sant.

Kari [02:03] Vi kan jo prøve.

(Trykker på grønt flagg. Tegner ikke trekant. Se Figur 20.)

Kari [02:07] Jaa! (Hendene i været) Det var det som skjedde.

Trine [02:08] Det ja …

Kari [02:09] Så vi må vri han 120 da.

Trine [02:14] Vi kan prøve.

Kari [02:15] Det dobbelte.

Trine [02:17] Det ble en trekant! (Se Figur 21.)

Kari [02:17] Jaa! (Hendene i været) Det ble det, det var bare det at vi glemte å slette alt.

Figur 20: Blokkode, ikke trekant

Figur 21: Trekant!

Som forventet programmerte elevene katten til å snu seg 60°, men jeg forventet ikke at Kari allerede før programmet hadde kjørt skulle forutse at det kanskje ikke ble en trekant. Siden hun alt hadde forutsett dette, ble hun glad selv når resultatet ikke viste seg å være en trekant. Elevene resonnerer seg fram til at katten må snu seg 120°. Jeg antar at tallet 120° dukker opp fordi Kari anvender annen matematisk kunnskap i denne nye situasjonen. Hun vet at nabovinkler er 180° og at 180° minus 60° er 120°.

Denne prosessen med å oppdage og rette på feil er et godt eksempel på *feilsøking*. Samtidig har elevene gjort vurderinger i forhold til hvor mange steg katten skal bevege seg, hvor mange grader den skal snu og hvor mange ganger dette skal gjentas. Elevene

har foretatt *evalueringer*. Visningsvinduet (Figur 21) viser resultatet hvor elevene har designet og *skapt* en trekant.

Figur 20 viser skriptet før de kjørte programmet for første gang. Figur 21 viser resultatet etter endring fra 60° til 120°.

Det neste elevene ønsket var å få programmet til å slette alt som var blitt tegnet i visningsvinduet hver gang det ble kjørt på nytt og få programmet til å tegne den samme trekanten på samme sted hver gang og ikke noe mer. Elevene eksperimenterte med blokkene og hvordan disse skulle pusles sammen. De testet programmet, men var ikke helt fornøyd og måtte gjøre nye vurderinger. De fant ut at de burde legge til en «penn av» blokk tidlig i *algoritmen*. Med andre ord har elevene *fiklet*, *feilsøkt* og foretatt *evalueringer*; alle prosesser knyttet til den algoritmiske tenkeren. Det tok ikke lang tid før malen for å lage flere regulære mangekanter nå var klar (Figur 22).

Figur 22: Malen

Så var det firkanten. Dette fikk de til uten problemer, de endret antall grader til 90, og antall gjentakelser til 4. Resultatet ble en firkant og de viste tydelig hvor fornøyde de var med seg selv. Elevene vet at alle sidekantene og vinklene i en regulær mangekant er av samme størrelse. Denne likheten utnytter elevene, de gjenkjenner *mønsteret* og endrer bare der de må. Dermed trenger de kun å endre på to tall i *algoritmen* sin for å få det ønskede resultatet.

Trine [03:21] Det ble en firkant. (Tydelig fornøyde) Eh, og så femkant. Da gjør vi … nesten det samme.

Kari [03:27] Er ikke det 115 eller noe? Jeg husker ikke hvor mye det er i en femkant. (Stillhet, tenker)

Elevene kjente på forhånd til størrelsen på de innvendige vinklene i en regulær trekant og firkant, men nå stod de fast med femkanten.

Kari [04:16] [???] på en sekskant er det bare til 60.

Trine [04:18] Ja, men på … femkant … for på en firkant er det 360, så da er det … vent, jeg må tenke … (ler) nei …

Kari [04:31] Fordi, når vi tok 60 i stad så ble det jo liksom starten på (viser med finger) et eller annet. Så hvis vi prøver å ta sånn rundt fem … ti, nei ikke femti, eh … Prøv å skriv 70.

Trine [04:43] 70?

Kari [04:44] Ja, det er 10 mer enn 60 … Det er matte.

Elevene forsøkte med 120°, som igjen resulterte i en trekant. De ble enige om at antall grader måtte være mindre enn det. De forsøkte å resonnere seg fram til hvor stor vinkelen måtte være og gjettet til slutt på 70. Dette var nærme, men ikke godt nok. Elevene gjorde flere feil, oppdaget feilene og prøvde igjen. De brukte *feilsøking*. Dette pågikk i flere omganger, elevene måtte fortsette og prøve igjen og igjen, de måtte *holde ut*. For hver gang de testet programmet måtte de vurdere om de skulle øke eller minke antall grader katten måtte snu, de foretok *evalueringer*. De kom stadig nærmere, og til slutt gikk det hvis katten snudde seg 72°.

Trine [06:03] Yes, vi klarte det!

Kari [06:05] Nå har vi lært det!

Trine [06:08] Sekskant.

Kari [06:10] Det var 60. Bare at vi må ta det 6 ganger tror jeg. Vi prøver det, for det føltes sånn i stad.

Trine [06:15] «Det føltes sånn i stad»?

Kari [06:17] Jeg følte at det var riktig.

Trine [06:19] Du bare føler det?

Kari [06:21] «Jeg føler det i meg». Ja, fordi. Ja! Fordi, ehm … En sekskant har dobbelt så mange kanter som en trekant, så det vil gi mening at de er dobbelt så store som en trekant.

Kari begrunner fremgangsmåten sin. Hun bruker *logikk* når hun analyserer og forutser at siden sekskanten har dobbelt så mange kanter som en trekant, vil det gi mening at de innvendige vinklene er dobbelt så store. Altså må katten snu halvparten av det den gjorde for å tegne trekanten. Trine arbeider godt sammen med Kari. Hun *samarbeider* og etterspør mer forklaring. Hun forstår Karis resonnement.

Elevene skrev inn 60 grader og 6 gjentakelser, men resultatet ble ikke slik som de hadde trodd. Det ble ikke en sekskant.

Figur 23: Ikke en sekskant

Kari [06:38] Det er sånn nærme nok da.

Trine [06:39] Det har syv kanter?

Tegningen ble ikke slik som forventet. På grunn av startposisjonen til katten og antall steg han beveger seg, gikk han utenfor nedre kant på visningsvinduet. Katten kom gående tilbake, men vinkler og avstander ble forstyrret uten at det var veldig synlig for elevene hvor forstyrrelsen skjedde. De godtok derfor at resultatet ble feil, og at 60° ikke kunne være riktig. De forsøkte videre med andre vinkler. Jeg hadde selv støtt på samme problem tidligere i mine egne utforskinger og kunne derfor hjelpe dem med problemet. Vi endret startposisjonen til katten og prøvde 60° på nytt. Da fungerte det.

Kari [08:05] Så vi hadde rett da egentlig?
(Prøver igjen, det virket)
[08:10] Derfor stoler man ikke på PC!

Neste oppgave var å tegne andre former. Elevene snakket om sirkler og stjerner og diskuterte hva slags stjerne de skulle tegne. Jeg ønsket at de skulle gå videre på syvkanten og legge grunnlaget for å løse oppgave 3 som jeg tenkte de kunne få til. Da de selv nevnte syvkanten, fulgte jeg det opp.

Kari [08:54] Eh, å ta liksom syvkant og sånn er jo bare å endre på grader greia. Så det blir jo egentlig litt kjedelig.

TC [09:01] Hvordan kan dere lage syvkanten da?

Kari [09:03] Vi må endre på, vi må gjøre sånn at han snur seg …

Trine [09:06] Endre der (peker) også i hvor mange ganger det gjentar.

Kari [09:09] Ja, mindre der, og større der.

TC [09:12] Hvor mye mindre må gradene bli da?

Kari [09:15] Nei, eh …

Trine [09:15] (ler) … Hmmm….

Kari [09:19] Så var det jo det da … En åttekant har eh …

Trine [09:25] Kan du det?

Kari [09:27] (ler) En åttekant?

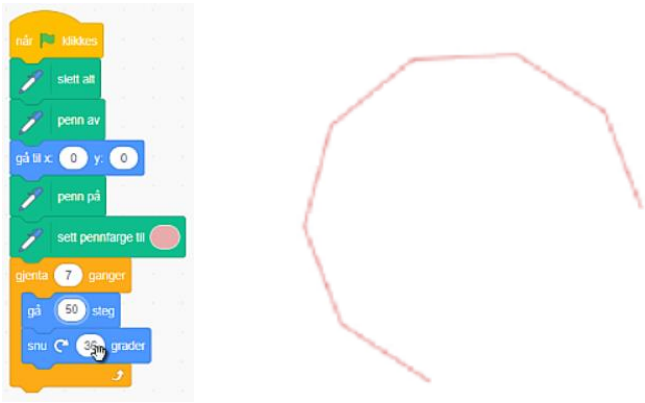
Trine	[09:28] Har det ikke bare dobbelt så m …
Kari	[09:30] Ja, prøv å ta 45 du. Ta 45.
Trine	[09:32] Okay. Ja, da er det 45 og så er det 8 ganger … For da er det … eller er det?

Elevenes første forsøk ble mislykket, fordi katten bevegde seg ut av visningsvinduet igjen. Etter å ha flyttet på startposisjonen og kortet ned på antall steg klarte de å lage åttekanten. De klappet og var svært fornøyde.

TC	[10:03] Så hvordan visste du at det skulle være 45 grader?
Kari	[10:06] Eh fordi, en trekant var 120 og …
Trine	[10:09] Ja! Og så …
Kari	[10:09] Da ble sekskanten 60 … så da … åttekant det er eh dobbelt så mange kanter som en firkant [???] alle er 90
Trine	[10:15] Og da er halvparten så mange … grader.
Kari	[10:18] Mhm. Halvparten av 90. … Så det går ikke an å lage en 16-kant, for da må det være komma (elevene trodde ikke at Scratch godtok desimaltall).

Elevene resonnerer ved hjelp av *mønstre*. Siden trekanten krevde at katten snur 120° og sekskanten, som har dobbelt så mange kanter, krevde en vinkel halvparten så stor, ville samme gjelde for firkanten og åttekanten. Firkanten krevde at katten skulle snu 90° og dermed må åttekanten, som har dobbelt så mange kanter, kreve en vinkel på halvparten; 45°. Elevene har oppdaget et *mønster* og *abstrahert* det. Å abstrahere vil si å trekke ut det generelle og fjerne unødvendige detaljer. Det spiller ingen rolle hvilken mangekant en starter med. Dobles antall sidekanter, skal den utvendige vinkelen halveres.

Nå var det på nytt syvkanten som skulle tegnes. Ettersom katten måtte snu 72° for å tegne femkanten, prøvde de 36° og syv gjentakelser. Dette ble ikke en syvkant.



Figur 24: Det ble ikke en syvkant, men ved å øke antall gjentakelser til ti, kunne de lage en tikant.

Kari	[11:22] Hvis vi gjør det flere ganger kanskje?
Trine	[11:24] (Begge ler) Da blir det ikke en syvkant.
Kari	[11:28] Da blir det en 10-kant.
Trine	[11:33] Tikant da … Tikant!

Elevene legger merke til at hvis en øker antall gjentakelser til 10 og katten snur 36°, vil de lage en ti-kant. Noe som stemmer overens med sin tidligere oppdagelse: For å lage en femkant måtte katten snu 72°, så hvis antall sidekanter dobles vil katten måtte snu halvparten av gradene.

For å spore elevene inn på flere mønster valgte jeg å systematisere funnene deres i en tabell. (På arket mitt lagde jeg små skisser av mangekantene. Elevene pirket på at de ikke var helt regulære.) De skrev selv på åttekanten og tikanten.

Mangekant	Antall grader	Antall gjentakelser
Trekant	120	3
Firkant	90	4
Femkant	72	5
Sekskant	60	6
Syvkant		
Åttekant	45	8
Tikant	36	10

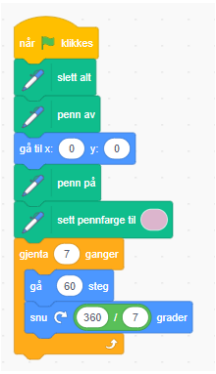
Tabell 6: Systematisere funn

Kari [15:11] Okay, jeg vet hva vi må gjøre … Prøve oss fram. Prøve og feile mye!
Trine [15:17] Se, det er det samme med denne og denne fordi det [???]
Kari [15:25] Så vi har ikke noen 3,5 kant her.
Trine [15:27] Nei, det har vi ikke.
Kari [15:29] Det er sykt kløne te egentlig … Men, vi kan jo egentlig bare prøve og feile sånn sykt mye … Hvis vi endrer den til 10 ganger, og så bare tar vi alt mellom 60 og 45 … Hvis du finner sånn ca midten. Det vil være 53. [???] 53, så ser vi om det fungerer.

Elevenes strategi for å løse problemet med syvkanten er å prøve og feile; en form for *feilsøking*, ettersom de ikke hadde en «tre-og-en-halv-kant» å basere seg på. Hadde de hatt en «tre-og-en-halv-kant» kunne de bare halvert antall grader som ble brukt for å lage den. For å lykkes med sin plan er de villige til å *holde ut,* «prøve og feile sånn sykt mye». Med oppgave 3) i tankene, valgte jeg å bryte inn i deres frie problemløsing.

TC [15:57] Hvis dere ser på katten ...
Trine [16:01] Ja, skal vi se på katten?
TC [16:02] … Når han beveger på seg.
(Gjør Felix synlig, tester 53 grader)
[16:05] Hvor mange grader …
Trine [16:07] (puster ut, oppgitt, mangekanten stemte ikke) 52? Jeg vet ikke, vent … (Retter oppmerksomheten mot TC)
TC [16:11] … går han totalt?
Kari [16:13] Litt over 360, ÅÅ! Åååå jaaa … (smiler til Trine) Vi må gjøre sånn at det blir 360.
Trine [16:17] For her er det …
Kari [16:18] 7 delt på … 360 delt på 7.
Trine [16:21] Her er det tre hundre [???] Her er det [???] åtti …
Kari [16:26] Nei, men du må jo gange det med 3 … Gange det med 4, gange det med 5, gange det med 6 … Det er 360 delt på 7 … Kan vi skrive på denne?

Etter denne lille «dytten» fra meg oppdaget Kari at «Antall gjentakelser» multiplisert med «Antall grader» i tabellen alltid ble 360°. Dermed kunne hun løse problemet med syvkanten ved å bruke divisjon. 360° dividert på 7 vil gi antall grader katten må snu. Kari har funnet likheter mellom alle mangekantene og kan bruke dette til å lage flere mangekanter. Hun har oppdaget nye *mønstre*. Elevene skulle til å utføre divisjonen for hånd, men jeg foreslo å bruke «operator»-blokker i Scratch til å gjøre utregningen. Elevene benyttet seg da igjen av *dekomposisjon*; først skrive kode for å beregne divisjonsstykket og deretter pusle det sammen med resten av koden (*algoritmen*). Da fikk de tegnet syvkanten og jeg gav dem andre halvdel av oppgavearket og ba dem gå videre til oppgave 2).



Figur 25: 7-kant

Poenget med oppgave 2) var å vise elevene hvordan programmet kan be brukeren om input og lagre verdien i en variabel. Elevene klarte raskt å lage programmet som stod på arket. Jeg antar at fargekodene gjorde det lettere å finne de riktige blokkene.

Kari [20:14] Det her er jo bare en lettere måte å tegne en syvkant på! … Eller er det enda mer komplisert kanskje? (Tester programmet)

Trine [20:20] «Hvor mange grader?»

Kari [20:41] Skriv 3,5!

Trine [20:42] Shhhh…

Kari [20:43] Ja, men jeg vil finne ut av hvordan en 3,5 kant ser ut … Nei, ja … Klar?

Elevene *fiklet* med koden i noen minutter; de eksperimenterte med ulike tall, men uten å helt finne ut av nøyaktig hva koden gjorde. Jeg ba dem uansett gå videre til oppgave 3). Målet med oppgaven var nådd for min del, nå ble det spennende å se om de klarte å bruke disse nye programmeringsblokkene til å løse et nytt problem. Oppgaven er «Klarer du å lage et program som ber brukeren oppgi antall hjørner, og deretter tegner en mangekant med det gitte antall hjørner?»

Kari [22:49] Her er det … ehm … liksom gradene. Han snur 360 delt på de gradene. Sånn at hvis du skriver 50, nei … 36, for det er greit. Så blir det jo.

Kor [23:08] Ti!

Kari [23:11] Så da gjentar han det 10 ganger. Og da går han 25 steg, så snur han, hvor mange, jo, 36 grader. Går 25 steg, snur 36 grader … 10 ganger.

[23:29] Så hvis vi … vi må først endre den [???] sånn at den heter hjørner. (endrer navn på variabelen fra grader til hjørner)

[23:57] F.eks. den med 7-kant. Da ender han jo opp med å måtte … Kanskje vi ikke trenger den (peker på variabelen som angir antall grader)? Nei, jo det gjør vi.

[24:07] Men da må vi ha sånn ekstra greie, må vi ikke det?

Trine [24:09] Sånn ekstra greie?

Kari [24:10] Ja, det må vi … Tror jeg … Jeg har ikke peiling.

Trine [24:13] Vi kan prøve … Men hvilken ekstra greie er det du tenker på?

Kari [24:17] Fordi, hvis vi legger inn sånn operator ting. Også tar vi …

Trine [24:20] Ehm, og da er det …

Kari [24:24] … liksom … For hvis vi lager en ekstra sånn variabel ting. Som heter, ehm, grader da. Ikke sant? Så sier den hvor mange grader den skal snu. Og de gradene er 36 delt på hjørner. Sånn at den snur ikke «hjørner» den snur «grader», ikke sant?

Kari tok føringen, men Trine er flink til å følge med på resonnementet og spør Kari om å utdype når hun ikke er tydelig nok. Elevene analyserer først koden fra oppgave 2) og forsøker å forutse hva som vil skje; de bruker *logikk*. De blir enige om hvilke endringer de må gjøre for å oppnå ønsket resultat. Elevene byttet navn på variabelen fra det opprinnelige programmet fra «grader» til «hjørner». De har gjort *evalueringer*; vurdert at et navnebytte var lurt. Da stod det at katten skulle snu «hjørner» grader, det gav ikke mening. De trengte litt hjelp med å opprette en ny variabel som de kalte «grader».

Kari	[25:23] Ok, den kan vi kalle grader.
	[25:30] Og grader er … da ehm (finner riktig operator blokk)… Gradene er 36 … 306, 360 mener jeg … delt på svaret … Sånn ikke sant, svar.
Trine	[25:49] Ja.
Kari	[25:52] Og den skal snu, på variabler, så skal den snu … grader. Ikke sant?
Trine	[26:00] Tror det. La oss prøve.
Kari	[26:02] Eh, bare endre den [???] Så blir det ordentlig.
	[26:08] Jeg vet ikke om dette kommer til å fungere. Skal vi prøve?
Trine	[26:10] Ja.
	(Tester)
Kari	[26:13] Ja, hvor mange hjørner? Okey, 3? Vi vet hvordan den skal se ut.
Trine	[26:17] Den ser ut som en trekant (fnis) … Wow, vi klarte å lage en trekant. Også står det der hvor mange grader (peker på skjermen). Wow!

Elevene har nå brukt det nye programmet til å lage en trekant. De hadde jo laget en trekant før, men denne gangen har programmet spurt dem om antall hjørner og beregnet antall grader katten må snu ut ifra svaret de skrev inn. Denne gangen er Trine mer begeistret enn da de lagde den første trekanten.

For å lage trekanten hadde elevene brukt en operator-blokk for å beregne «360/hjørner» som stod som antall gjentakelser i gjenta-blokken. Dette gjorde at programmet av og til fungerte slik de ønsket, men noen ganger ikke. Elevene utforsket og eksperimenterte med ulike svar til programmet. De *fiklet*, endret svarene sine, men også i selve *algoritmen* endret de på hvor mange steg katten skulle bevege seg før den snudde. Dermed kunne de også endre størrelsen på mangekantene. Å lage mangekanter med få antall hjørner gikk greit så lenge kvotienten «360/hjørner» ble et tall høyere enn antall sidekanter (katten tegner de samme linjestykkene flere ganger). Men når elevene prøvde å lage mangekanter med 42 eller 56 hjørner fungerte det ikke.

Figur 26: Forsøk på 56-kant

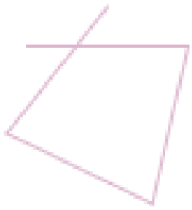
TC	[28:00] Hvis dere kikker litt på den derre gjenta klossen deres.
Trine	[28:04] Gjenta …
Kari	[28:05] Ja, den …
TC	[28:06] Hvor mange ganger skal han gjenta?
Kari	[28:07] 360 delt på hvor mange hjørner det skal være … Men det trenger vi jo ikke å ha.
Trine	[28:12] Nei, vi kan ta vekk den …
Kari	[28:16] Gjenta sånn 850. Nei, vent litt …
Trine	[28:20] 850 ganger (ler)
Kari	[28:21] Gjenta hjørner, ikke sant?
Trine	[28:23] Ja. Åh!

Elevene fant ut av hva de kunne skrive i gjenta-blokken. Ved å analysere hva de individuelle blokkene gjorde, og forutse hva de ønsket at skulle skje, brukte elevene *logikk* til å bli enige om at katten skulle gjenta blokkene i løkken «hjørner» ganger, og endte opp med et program de ble fornøyd med.

Figur 27: Elevenes løsning av oppgave 2) og 3)

Ved hjelp av deres nye program tegnet elevene først en 56-kant og deretter en 42-kant. Disse mangekanten ble så store at elevene måtte eksperimentere med ulike antall steg katten skulle gå, de *fiklet* seg fram til 12 steg. Da fikk mangekantene plass i visningsvinduet.

Elevene avsluttet timen med å endelig kunne tegne en «tre-og-en-halv-kant».



Figur 28: Tre-og-en-halv-kant

Elevene har i løpet av timen vært innom alle arbeidsmåter og nøkkelbegrep knyttet til den algoritmiske tenkeren. I begynnelsen er det mye fikling for å skape algoritmene, dette lager en setting hvor elevene naturlig går på jakt etter mønster og bruker logikk og evalueringer for å gjøre tilpasninger. Elevene har definitivt arbeidet som programmere.

4.2.2 Scratch – Lineære funksjoner

I denne klassen var det kun to elever som hadde fått underskrift på samtykkeskjemaet, begge var gutter. I denne oppgaven kaller jeg dem «Erik» og «David». Det var en annen stemning i denne klassen under introduksjonen sammenlignet med forrige klasse. Elevene virket mer umotiverte og var lite engasjert i den felles introduksjonen. Jeg oppfattet at de hadde negative forventninger til programmering som en del av undervisningen.

Etter introduksjonen ble Erik og David med meg ut på et grupperom og fikk oppgavearket (8.4 Oppgaveark: Scratch – Lineære funksjoner). Deres holdninger var ikke representative for klassens, disse to kom til liv da vi forlot klasserommet. Jeg ba dem tenke høyt mens de løste oppgavene og gjerne fortelle hva de forventer å se før de kjører programmet sitt. Begge elevene var allerede kjent med programmeringsspråket Scratch. De visste hvor de skulle lete etter blokkene de ville bruke og hvordan de skulle pusle dem sammen for å lage et program. De første oppgavene gikk derfor unna raskt, men ikke helt uten problemer.

Det første de gjorde var å finne fram «når grønt flagg klikkes»-blokken. Deretter valgte de å opprette to variabler, en de kalte for «nå» og en de kalte for «født». De fikk programmet sitt til å stille to spørsmål: Først «Hvor gammel er du?», dette svaret ble lagret i variabelen de kalte «født». Deretter spurte programmet «Hvilket år er det?», dette svaret ble lagret i «nå»-variabelen.

Erik	[02:56] Sånn, hvilket år er det, spør om det. Det blir jo et svar, så må vi på variabler igjen. Så setter vi «nå» til det året. … Eh, der ja. … Okay, og deretter svarer hvor gammel brukeren vil være om 10 år. … Da har vi fått de to svarene. … Så, vi tester hvordan det er så langt, start. «Hvor gammel er du?» så skriver du
David	[03:35] 15
Erik	[03:36] «Hvilket år er det?» 2019. … Okay, alt fungerer så langt. … Men så… er det på… sett sammen… men først må vi ha den der (finner blokken «si») … Da må du bare ta «om ti år er du».
David	[04:26] Vent litt, stor «m», nei…
Erik	[04:29] Stor «o». Også, om ti år er du «banan», nei det er du ikke. Men, skal vi se… Om ti år er du, da må du ta året nå minus noe… om ti år er du, skal vi se … Det må bli 2019 minus 2004, for da blir det et positivt tall i stedet for negativt.
David	[04:54] Mhmm.

Elevene er vant med å prøvekjøre programmene sine for å sjekke at de fungerer som tenkt. «[…] vi tester hvordan det er så langt […] alt fungerer så langt.» De får da muligheten til å oppdage eventuelle feil og rette på dem; de bruker *feilsøking*.

Den korte seansen over viser hvordan elevene *dekomponerer*; de bryter problemet ned i mindre deler, løser dem først, og deretter pusler dem sammen for å skape *algoritmen*. De får programmet til å først stille spørsmål til brukeren hvor svaret blir lagret i en «svar»- variabel. Katten skal si noe, men før den kan gjøre det må de sette sammen noen flere blokker. De velger blokken «sett sammen» og setter sammen setningen «om ti år er du» og en beregning «nå» minus «født» pluss ti. Nå har de løst delproblemet om hva katten skal si og bruker denne løsningen sammen med «si»-blokken som igjen pusles sammen med resten av koden.

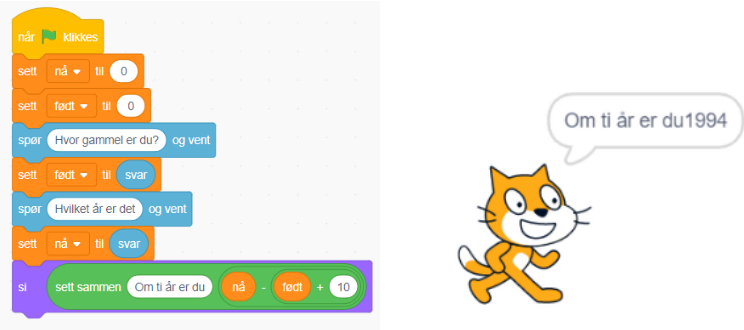
I oppgave a) blir elevene bedt om å lage «et program som spør brukeren hvor gammel hun er og som deretter svarer hvor gammel brukeren vil være om 10 år». Elevene fokuserte på eksempel 2 fra oppgavearket da de skulle løse oppgave a). De stilte to spørsmål: «Hvilket år er det?» og «Hvor gammel er du?». Samtidig opprettet de to variabler som i eksempel 2; «nå» og «født». Ved å ta utgangspunkt i eksempel 2 med to variabler og to spørsmål har de gjort det vrient for seg selv.

David [05:33] Da prøver vi… 15

Erik [05:37] Jeg håper ikke vi har glemt noe. … alt skal være riktig.

[05:50] Om ti år er du 1994

(Latter!)



Figur 29:Første forsøk oppgave a)

Programmet spør «hvor gammel er du?», men svaret lagres i en variabel som heter «født». Elevene har rast av gårde i stor fart, funnet blokker og laget variabler, men ikke tenkt nøye nok over hva oppgaven faktisk spør om og om de svarer på dette.

Poenget med oppgave a) og b) var i utgangspunktet å gjøre elevene kjent med Scratch. Hvis elevene hadde brukt eksempel 1 som mal for oppgave a), ville de sannsynligvis ha fått det til med en gang. Heldigvis, vil jeg si, brukte de «feil» eksempel som utgangspunkt og endte dermed opp med et problem som måtte løses. Fordi elevene ikke fikk et ønsket resultat blir de nødt til å vurdere hva som er galt; de må gjøre *evalueringer*.

Erik [05:54] Åja, det er jo fordi du har… må vi plusse på 10 først? Så da tar du… «Om ti år» kan du legge til mellomrom (mellom tekst og tall). … [???] fordi, den vil jo først plusse sammen, så vi tar minus etterpå. … men da må vi… skal vi ha det i en egen snakkeboble? Eller må vi til med parentes her? … nei

David [06:36] Shii, jeg vet ikke. … Kan vi ikke bare bytte på de?

Erik [06:44] Tror ikke det går, for du tar født + 10 år … JO!

David [06:47] Hva er det det står her? (ser på arket) Vi må bare bytte…

[06:50] Her, her står det ikke noe pluss, men. Eller nei, ja, vi må jo plusse

Erik [06:54] Det er eksempelet, de gir oss jo ikke svar på oppgaven. … født pluss 10 minus … okay, hvis vi tar dette … det blir et negativt tall, så disse to må bytte plass. Hvis vi tar «nå» pluss 10, ja det skal jo stemme.

David [07:14] Skriv 10

Erik [07:16] For du tar jo nå, 2019, 2029 blir det jo da, minus 2004 som vil være noe… Det er ikke … eh…

David [07:27] Men bare prøv.

Erik [07:30] Hvor gammel er du, skriv 15. Oi, begge trykket samtidig.

David [07:37] Åja…
Erik [07:40] Hvilket år er det? 2019.
David [07:45] Hva i? … Vent litt. …

Deres første antakelse var at det var noe galt med regnerekkefølgen. «nå» - («født» + 10) burde heller vært («nå» + 10) - «født». Elevene bruker *logikk* når de analyserer regnestykket på denne måten og forutser et resultat som gir mening for dem. Men når de kjører programmet får de «Om ti år er du 2014» til svar. Fortsatt ikke det de forventer.

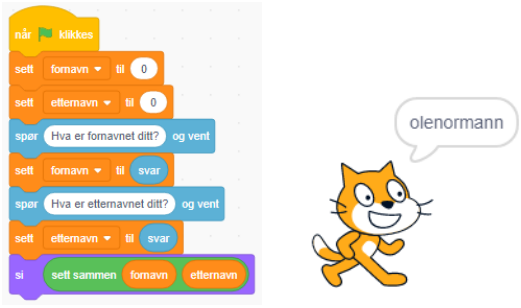
(Erik legger hodet ned på bordet)
Erik [07:57] Jeg innså hva vi gjorde feil.
David [07:58] Å ja.
Erik [08:00] På «født» skrev vi 15 år.
David [08:03] Å ja! … Skal vi forandre på den da?
Erik [08:08] Ja, nei, du må… Vi må bare skrive «hvor gammel er du?» 2004
David [08:13] Å ja (ler)
Erik [08:14] Hvor gammel er du?, vi greide å forvirre oss selv. … Det er jo genialt av oss.
(svarer 2004 på «Hvor gammel er du?»)
Erik [08:21] Så det funket!
(David klapper)

I resonnementet sitt sier Erik at programmet må ta årstallet nå, 2019, legge til 10 slik at det blir 2029, og deretter trekke ifra året da brukeren ble født, 2004. Gjennom resonnementet oppdager Erik hvor feilen ligger. Spørsmålet som stilles er et annet enn det navnet på variabelen viser til. Ved å svare 2004 på spørsmålet om alder, virker programmet som tenkt. De forvirret seg selv, men løste problemet allikevel.

David [08:25] Skal vi forandre på verdi… variabelen?
Erik [08:29] Vi tar heller bare «Hvilket år er du født?» isteden.
David [08:32] Hehe, så vi bare unngår problemet.

Begge elevene ser hva problemet er og har forslag til hvordan koden kan rettes på. De gjør en *evaluering* og endrer spørsmålet til «Hvilket år er du født?». De sjekker at alt fungerer og går så videre til oppgave b).

Elevene bruker under fire minutter på å bli ferdig med oppgave b).



Figur 30: Oppgave b)

De ser at det mangler mellomrom mellom fornavn og etternavn og eksperimenterer med ulike forbedringer, de *fikler* seg fram til en løsning. Hvis brukeren trykker på

mellomromstasten etter å ha oppgitt sitt fornavn, fungerer det. Det foreslås også å lage en variabel som kun inneholder et mellomrom.

Erik [13:30] Sånn, nå ble det riktig, men vi kunne også egentlig bare lagt til en variabel som bare hadde vært et mellomrom, så hadde den kommet automatisk. Tror jeg hadde vært bedre. Men den funker den nå da

Elevene går videre til oppgave c) hvor de skal følge en link og fylle inn det som mangler for å lage den opprinnelige funksjonsmaskinen min (https://scratch.mit.edu/projects/341400228/editor/). De leser gjennom oppgaveteksten og kikker på koden.

Erik [14:44] Men det er hvertfall… så er det jo… ja, han vil gå bakerst og bli mindre, så vil den skjule, nei, alt det der… og så vil den gå der, bortover sånn, bli vist, spør «velg et tall», er det startkoden, alt det der, og da vil den ende opp der. Sånn at den blir lagt bak laptoppen og alt det der. Det trenger vi ikke fokusere på, men. Spør «velg et tall og vent». Da setter vi input til det svaret vi får.

David [15:15] mhm.

Her analyserer Erik blokkene og hvordan de er satt sammen og bruker *logikk* til å finne ut av hva som mangler i de tomme boksene. Han velger bort delene av *algoritmen* som har med flyttingen av katten å gjøre, «Det trenger vi ikke fokusere på».

Erik [16:49] Åja, det er sant det for den gikk fra over der og kom ut med svar. Svar. Så kommer den ned der, ikke sant, output vil være svaret, eller vi kunne også brukt input der, … gange to pluss en, så … sånn…

David [17:23] Og så si …

Erik [17:25] Da vil den si…

David [17:26] Output.

Erik [17:31] Så vil den… jo, jo, vent litt. Gå til kanten, sett sammen, sett sammen. Da blir det svaret…

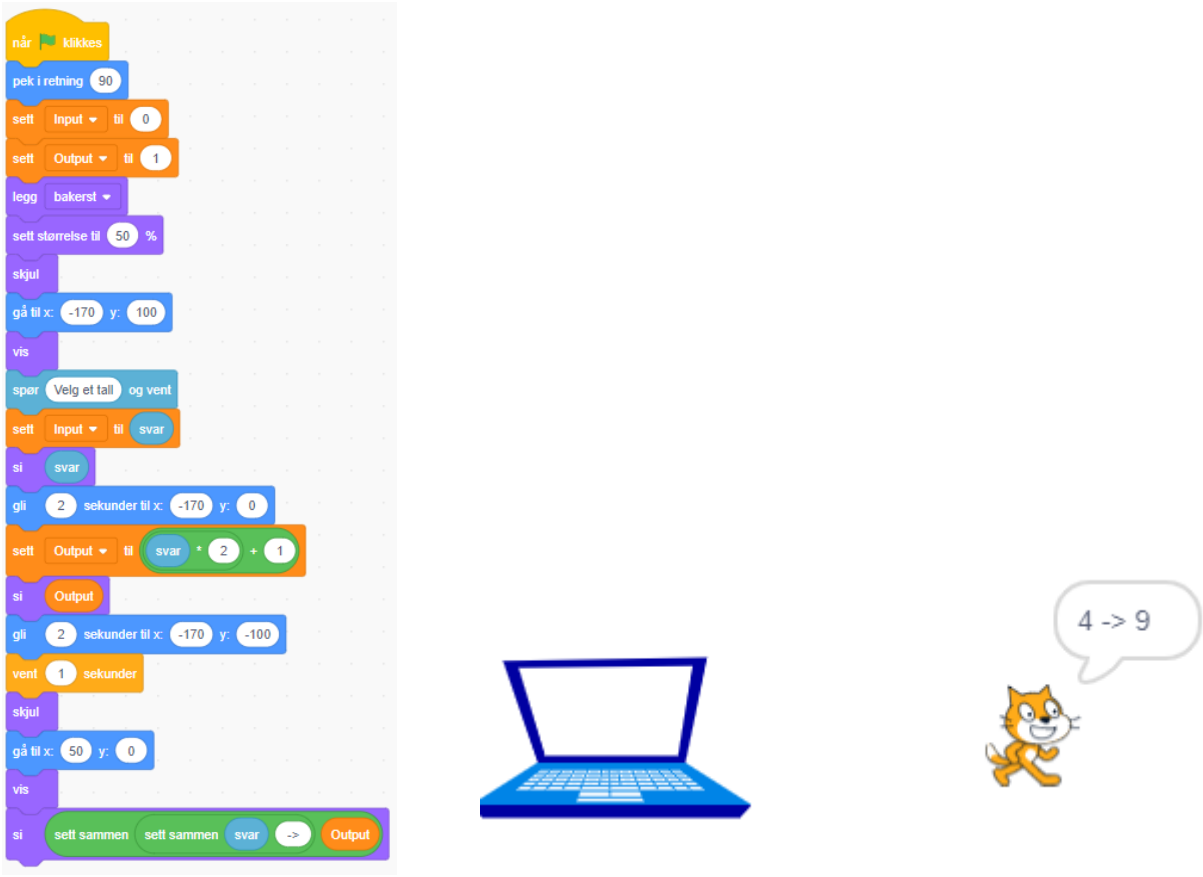
David [17:42] Nei, nei, nei… Det vi sa først.

Erik [17:45] Ja, det er svaret, svaret vil forbli der, fordi output…

David [17:48] Ja, svaret og så output, ja ja ja.

Erik [17:56] Skal vi teste den? … Velg et tall.

Elevene deler tankene og forslagene sine med hverandre, de *samarbeider*. De viser at de forstår hvilke verdier som er lagret i de ulike variablene; «input» tilordnes verdien «svar», så begge disse variablene inneholder den samme verdien, så lenge programmet ikke spør om noe mer fra brukeren, «Svar […] eller vi kunne også brukt input der». «Output»-variabelen tilordnes en verdi basert på en utregning med svaret fra brukeren og dette skal katten si til slutt. Resultatet er en funksjonsmaskin klar til bruk.



Figur 31: Funksjonsmaskin mal

I siste oppgave på oppgavearket skal elevene gjøre sine egne endringer og lage egne versjoner av funksjonsmaskinen.

David	[18:29] Oh, hva er det vi skal gjøre da?
Erik	[18:30] Det hadde vært mye gøyere å holde på med dette her i matten.
David	[18:32] Ja, gange… Hvor mye skal vi ta gange?
Erik	[18:34] Her kan vi bare ta…
David	[18:35] Ja, bare forandre på den.
Erik	[18:36] Vi kan også ta deling da? … og forandre på denne? … Men hvis vi tar, la oss si, gange med …
David	[18:46] 5
Erik	[18:46] 3 pluss 5 eller et eller annet sånn.
David	[18:48] Ja
Erik	[18:48] Bare skriv tilfeldige tall… Ikke så høyt men… 5 pluss … sånn … så må vi teste, det skal jo egentlig funke men.

Elevene forstår med en gang hvor i *algoritmen* de må gjøre endringer. De fjerner alle de unødvendige detaljene, og bedriver da med *abstraksjon*. Det eneste stedet de må gjøre endringer er der hvor verdien til «output» variabelen blir beregnet. Ved å endre beregningen til noe selvvalgt har de *skapt* sin egen funksjonsmaskin.

David	[19:02] Så for eksempel 10
Erik	[19:04] Da skal svaret bli 53.
David	[19:06] Gange, ja, 53. (Det stemte)

Erik [19:08] Da funker det. Da er egentlig alle oppgaver gjort… ja.

Figur 32: Oppgave d)

Jeg ba elevene teste programmet med et nytt tall. Verdiene, input og output, skrev jeg ned på et ark. Hadde disse elevene løst oppgavene i hel klasse, ville de hatt muligheten til å se hvilke andre funksjonsmaskiner de andre elevene hadde skapt. Siden de var alene på et grupperom, ba jeg dem lage enda en versjon selv. De foreslo å prøve divisjon istedenfor multiplikasjon, da utforsker de muligheter og eksperimenterer med løsninger på eget initiativ; de *fikler*. Totalt hadde vi nå tre funksjonsmaskiner og fikk testet noen forskjellige verdier.

Funksjonsmaskin 1	Funksjonsmaskin 2	Funksjonsmaskin 3
(2 , 5)	(10 , 53)	(10 , 8)
(4 , 9)	(20, 103)	(100 , 53)
(5 , 11)	(6947 , 34738)	

Tabell 7: Tre funksjonsmaskiner

Elevene sa de var kjent med GeoGebra, så jeg ba dem tegne inn verdiene fra funksjonsmaskin 1 som punkt i koordinatsystemet.

David [23:36] fire komma ni.

Erik [23:43] Jeg tror jeg vet hva som kommer til å skje

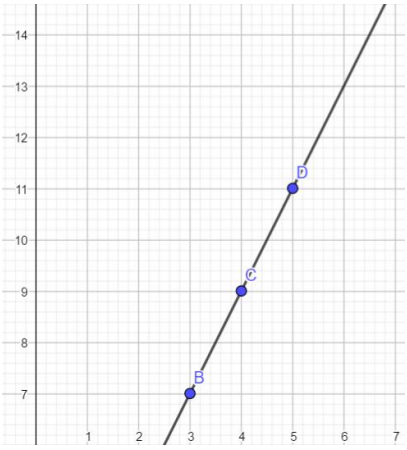
David [23:46] Ja. Fem komma elleve.

TC [23:52] Og hva er det du tror kommer til å skje, sa du?

Erik [23:54] At vi fortsetter med samme mønster oppover. Istedenfor å bruke den maskinen der, kunne du bare sett at du skal en bort og to opp, og da ville … svaret… du ville hatt 6 og fått ut 13 (peker på koordinatsystemet). Så da vil du kunne brukt GeoGebra eller koordinatsystemet til å finne ut hva svaret ville vært før du hadde tastet inn i maskinen. … Det er vel et verktøy, hvis du for eksempel hadde tatt den linja så ville den gått oppover der…

Erik tegner ei linje mellom to av punktene i koordinatsystemet og bruker den linja til å forutse verdier. Han kikker etter sammenhenger og har funnet et *mønster*. Ved hjelp av mønsteret bruker han *logikk* til å forutse hva som kommer til å skje. Han beskriver lineær stigning med ordene «en bort og to opp».

Erik [24:44] Si at vi for eksempel tok 12, da ville det blitt 25. (Bruker linja til å forutse verdier) Jeg kan se svaret her ut i fra den linja.



Figur 33: Lineære sammenhenger

Jeg ba elevene tegne inn punkt fra deres første funksjonsmaskin, funksjonsmaskin 2.

David [25:10] ti komma femtitre…

Erik [25:14] da vil vi vel også kunne forutse hvilke tall maskinen vil regne ut.

David [25:28] tjue komma hundreogtre

Erik [25:21] Hvorfor tok vi egentlig så høye tall? (ler) … tjue komma hundreogtre, var det ikke det?

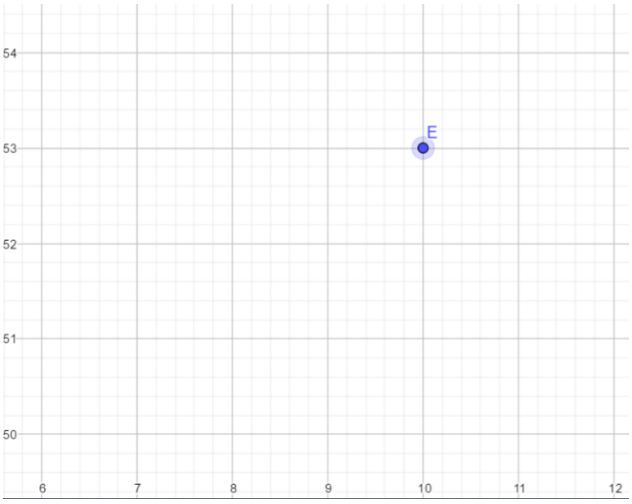
David [25:25] Ja.

(Blar og zoomer i grafikkfeltet for å finne disse punktene)

Erik [25:57] Hvis vi hadde tatt en linje mellom de to punktene så vil vi kunne valgt tall og der de… ja… vi kan se på et eksempel her.

(zoomer inn slik at punktet E er tydelig markert på rutenettet).

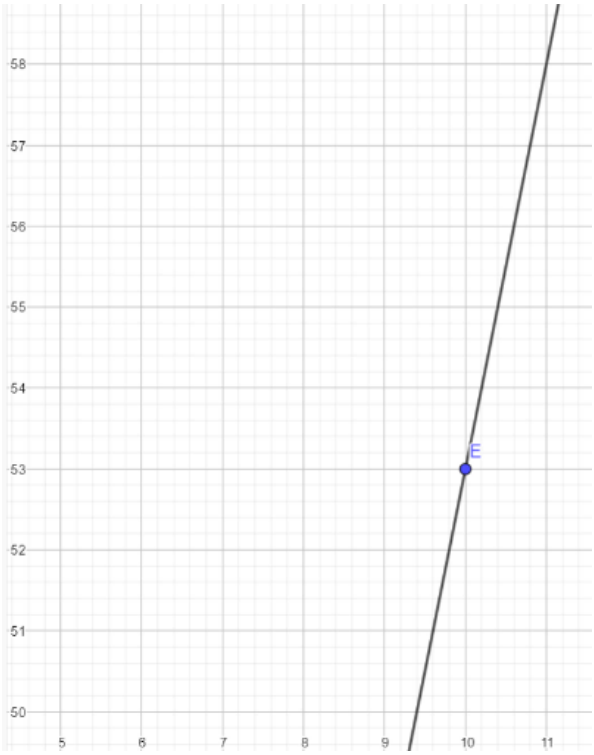
Erik [26:12] Det krysningspunktet der. Så ville vi f.eks. kunne tatt 11 in og funnet ut at det blir 54 uten å trenge å bruke maskinen.



Figur 34: Punktet E

Erik [26:22] Eller det kan være at linja blir litt annerledes … Jeg kan trekke en linje så blir det litt lettere å se, for det blir sannsynligvis litt feil. … Eller mest sannsynlig bli et annet tall.

David [26:40] Det burde bli 58.



Figur 35: Grafen gjennom E

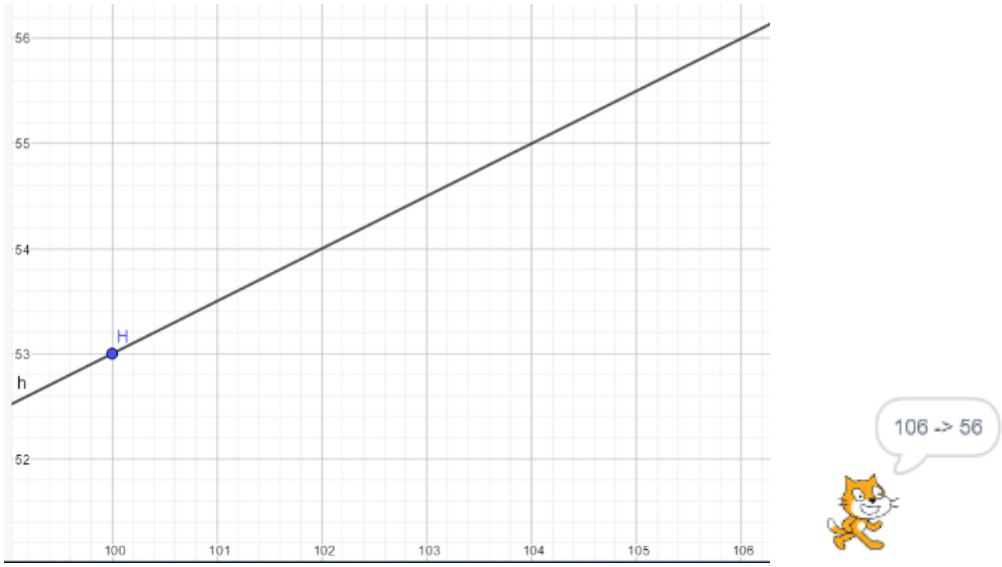
Når Erik forstørrer grafikkfeltet ser han punktet E (10 , 53) tydelig som et punkt på rutenettet. Han antar da at ett hakk bort, ett hakk opp fortsatt vil fungere og at (11 , 54) også vil være et punkt på den rette linja han skal tegne inn. Mens elevene skal til å tegne linja mellom punktene, ombestemmer Erik seg. 11 inn gir kanskje ikke 54 ut. De tegner linja og ser at 11 inn gir 58 ut, som stemmer med det David sa.

Videre tegner de inn de to punktene de har til funksjonsmaskin 3.

TC	[27:31] Hva tror vi på forhånd kommer til å skje?
David	[27:34] At det blir samme linje. Rett linje.
Erik	[27:38] Det går en linje mellom punktene, og igjen vil de krysse der, og da kan vi si… hvis det for eksempel hadde vært at 12 hadde blitt til 54 så kunne vi finne det ut uten å måtte taste det inn i maskinen eller regne det ut. Fra den dataen vi allerede har kunne trekke en linje og funnet ut hvilke tall vi ville fått. Vi kunne forutsi hva maskinen ville svart.

Dette er en *abstraksjon*; Erik forklarer at den rette linjen representerer alle verdiene vi kan få ut av funksjonsmaskinen. Vi trenger ikke taste inn i funksjonsmaskinen, vi kan «forutsi hva maskinen ville svart».

Erik	[28:23] 106 ville blitt til 56. … Og det kan vi teste med maskinen hvis vi skriver inn 106. (endrer vindu til Scratch, tester)… Jepp, det ble riktig.



Figur 36: Graf og funksjonsmaskin

Til nå har elevene pekt på heltallsverdier, så jeg spurte om hva som skjer hvis vi velger å sende et desimaltall inn i maskinen? Ved å forstørre rutenettet i GeoGebra slik at aksene viste desimaltall klarte de å forutse dette også. De byttet til funksjonsmaskinen i Scratch for å sjekke at deres antakelse stemte.

TC	[31:18] Da lurer jeg på om dere kan lage funksjonsmaskinen på en sånn måte at resultatet, altså koordinatene, ikke blir ei rett linje? Er det mulig eller er det umulig?
David	[31:30] Hmm…
Erik	[31:33] Det vil være umulig for på et eller annet tidspunkt så vil de linjene treffe hverandre for de er ikke…. Ja, hva heter det igjen?... De er ikke parallelle, de blir bare på rett sånn (viser med hendene) og da vil du på et av krysningspunktene så vil du på et eller annet vis treffe…
David	[31:48] Vi kunne jo se at her er det sånn 2 tall (peker på dividert på 2 i koden) også kunne det ha vært sånn tilfeldig til hvilken det byttet…
Erik	[31:55] Det kunne variert på den…
David	[31:59] Men da er det tilfeldig, da er det ikke noe sånn rytme, eller.

For at funksjonsmaskinens tilordning ikke skal resultere i en rett linje, foreslår elevene å opprette en ny variabel som inneholder et tilfeldig tall. Da vil det ikke være noen «rytme» i punktene. Dette var en løsning jeg på forhånd ikke var forberedt på. De har jo helt rett; hvis vi oppretter en variabel som tilordnes et nytt tilfeldig tall hver gang programmet kjøres og vi bruker denne variabelen i beregningen, vil det ikke være noe «rytme». Men da risikerer vi også at det samme tallet inn, vil gi forskjellige verdier ut fra gang til gang, altså vi har ikke lenger med funksjoner å gjøre. Jeg etterspurte ikke mer i forhold til den løsningen.

Elevene viser at de behersker bruken av variabler i programmering og forstår hvordan de vil påvirke representasjonen av funksjonsmaskinen i GeoGebra. Overgangen fra funksjonsmaskin i Scratch til koordinatsystem i GeoGebra har gått smertefritt.

Ettersom Erik hadde nevnt parallelle linjer, fulgte jeg opp dette og spurte om de kunne lage en ny funksjonsmaskin hvor den resulterende linjen ville være parallell til en av dem de hadde fra før.

Elevene forsøker å lage en parallell linje til funksjonsmaskin 3, hvor input verdien ble dividert på to og fikk lagt til 3. De tegner en linje i GeoGebra som er parallell med denne, men sliter med å finne et uttrykk som vil passe.

Erik [35:25] Ja, parallell. Da måtte 10 bli til 10, 14 bli til 12, har vi noe mønster ut av det? … 6 til 8… På et eller annet tidspunkt vil det bytte retning, så jeg tror ikke det er helt mulig men.

David [35:44] Vi må bare finne forholdet til de to tallene.

Erik [35:48] Problemet er at det varierer, for du ser at 2 blir til 6, 4 blir til … nei, det blir desimaltall, må zoome ut da. … 2 blir 6, 3 blir 6,5 ca. , 4 blir til 7, 6 blir til 8, 8 blir til 9, 10 blir 10, 12 blir til 11, nei vent litt… pluss fire, pluss 3, pluss to, pluss 1, pluss 0… Der blir det minus, så det ville byttet, 10 er midtpunktet. Det ville byttet fra pluss til minus, så det ville vært litt vanskelig, men det kunne kanskje vært mulig, men det tror jeg ikke vi får til akkurat nå.

David etterspurte «forholdet» mellom x- og y-verdiene. Erik fant differansene mellom verdiene og oppdaget et *mønster*.

x- verdi	2	4	6	8	10	12
differanse	4	3	2	1	0	-1
y-verdi	6	7	8	9	10	11

Tabell 8: Erik fant mønster

Erik kalte 10 for «midtpunktet», da var differansen lik 0, men deretter ble det «minus». Dette fant de ikke ut av og forsøkte seg heller på en ny linje, parallell til en av de andre funksjonene.

David [37:32] Skal vi begynne fra 0?

Erik [37:33] Vi kan ta den her. 1 blir til 1, 2 til 2, 3 til 3, nei her blir det bare å gå… opp til to… Hvis jeg trekker linja. Den bli til den, en bort og to opp…

David [37:58] Kanskje hvis vi går på Scratch, også deler de i to, tallene to, da blir det jo…

(Erik fortsetter å tegne den linja han startet på)

Erik [38:20] Der har vi for eksempel den. For jeg sa 1 blir til 2 der. 2 blir til 4.

David [38:26] 3 blir til …

Erik [38:28] 3 blir til 6, så da ganges det med 2, så den vil det være mulig å [???]

(Endrer koden i Scratch slik at funksjonsmaskinen bare multipliserer med to uten å legge til noe)

I GeoGebra har elevene tegnet en linje som er parallell med funksjonsmaskin 1 (hvor input verdien ble multiplisert med to og fikk lagt til 1) og som går gjennom punktet (0 , 0). De observerer at input verdien til funksjonsmaskinen må multipliseres med 2 for å få output verdien som stemmer med linjen. Dette lager de i Scratch og har da en funksjonsmaskin som kan representeres med en rett linje som er parallell med funksjonsmaskin 1. Igjen så er det *mønstrene* som er sentrale for løsingen av oppgaven; «en bort og to opp» sier Erik.

Jeg utfordrer dem videre med å spørre om det er mulig på forhånd å vite om linjene blir parallelle hvis de kjenner uttrykkene. Er det noe felles med uttrykkene?

David [40:11] Begge er ganger to.

Erik [40:17] Bare der har vi tatt min…, vi har tatt bort 1, så da blir det flyttet et hakk ned. Egentlig så ville linjen gått sånn, men siden vi flyttet den et hakk ned vil den forskyve seg bortover her.

TC [40:30] Så du mener at du kan lage enda en parallell linje?

David [40:32] Ja.

Erik [40:32] Mhm. Da kan vi ta, i stedet for, pluss 1 da tar vi pluss 2.

David [40:37] Mhm.

Elevene endrer uttrykket i Scratch igjen, slik at input verdien nå blir multiplisert med 2 og lagt til 2. De tester to ulike verdier, bruker tallparene som koordinater i GeoGebra og tegner en linje gjennom disse to. Linjen ble som ønsket parallell med de andre to linjene.

Erik [42:43] Så vi beholder den der, ja, ganging og deling, men vi forandrer på pluss og minus. Og når det er pluss, ja, det går oppover (linja forskyves oppover), eller om det går nedover, eller, tallet som blir plusset på. Så vi kunne, der ser du at det er minus, vi kunne for eksempel tatt pluss 5, da ville den ha flyttet seg 5 bortover sånn.

Erik forklarer at ettersom variabelen alltid blir multiplisert med 2 og de bare endrer på det som bli lagt til, eller trukket ifra, vil linjene forbli parallelle, men forskyves opp eller ned i forhold til hverandre.

TC [43:15] Eh… Hva tenker dere om koblingen mellom matematikk og programmering?

Erik [43:21] Det er gøyere med programmering og matte, enn bare matte.

David [43:27] Ja.

Erik [43:27] For i programmering da kan vi jo ha sånn der, lage sånn der maskin og så kan du prøve å finne ut av om det er noen sammenheng, alle de forskjellige svarene, og om du klarer å lage maskin som bare flytter litt på forskjellige svarene som blir gitt. … I stedet for å bare sitte og holde på med masse teori, så kan vi jo sitte og, jobbe med sånn der litt mer praktiske oppgaver og du får for eksempel den her oppgaven hvor du skal lage en maskin, så skal du finne ut hva er sammenhengen mellom det her og om du kan regne det ut på en eller annen måte uten å bruke maskinene, om du kunne se et mønster.

Elevene har likt å arbeide med programmering i matematikktimen. Erik peker på muligheten til å *skape*, *fikle* og se etter *mønster* som deler av sin begrunnelse. Selv om funksjonsmaskinen er virtuell, snakker Erik om å lage maskinen og kaller det en mer praktisk oppgave. Han har *skapt* maskinen. Han nevner alle de forskjellige svarene funksjonsmaskinene gir, disse verdiene har de fått ved å utforske og eksperimentere. De har *fiklet* med *algoritmen* for å finne sammenhenger og avdekke *mønster*.

Elevene har i løpet av timen vært innom alle arbeidsmåter og nøkkelbegrep knyttet til den algoritmiske tenkeren. De bruker gjennom hele opplegget feilsøking som en naturlig del av sin problemløsing. Etter at oppgavene på arket er gjennomført og elevene får oppgaver muntlig blir de eksponert for større utfordringer og må lete etter mønster og bruke logikk og evalueringer for å løse dem. Elevene har arbeidet som programmere.

4.2.3 p5.js – Sannsynlighet

Som forklart tidligere, fikk jeg ikke gjennomført et oppgavebasert intervju med dette opplegget. Jeg fikk derimot gjennomført det i min egen matematikklasse og vil her skrive kort om hvordan det gikk. Opplegget ble gjennomført i en periode hvor skolen var på rødt smittevernsnivå, noe som betyr at klassen var delt i to og jeg gjennomførte opplegget da to ganger. Hovedfokuset var selve opplegget og hvordan det ble tatt imot.

En skulle kanskje tro at ettersom det er min egen klasse burde elevene allerede være godt kjent med programmering, men dette var ikke tilfelle. Elevene fulgte på dette tidspunktet fortsatt LK06, programmering var ikke en tydelig del av læreplanen. Jeg vet de har arbeidet med noen «kodetimer» i andre fag tidligere, da med blokkbasert programmering, og seks av elevene har tidligere år hatt programmering som valgfag.

Jeg gjennomførte introduksjonen som planlagt: Først gjennomførte vi et «hesteløp» med én terning. Elevene gjettet på hvilken hest de trodde skulle vinne og fulgte spent med på programmet som kjørte på tavlen. Deretter fortalte jeg at vi skulle lage dette programmet fra bunnen. Elevene fikk en kort introduksjon til p5.js og dens tre hovedvinduer. Så fulgte jeg opplegget som forklart i kapittel 4.1.3 p5.js – Sannsynlighet.

Underveis i introduksjonen kom ikke elevene med noen spørsmål. Jeg stilte noen spørsmål av typen: «Hva bør skje nå?». F.eks. etter at vi hadde laget en terning, hva skal vi gjøre med den? Jo, vi må kaste den. Hva skjer hvis den viser en 1-er? Jo, vi må få hest1 til å flytte på seg. På denne måten prøvde jeg å få elevene med på hva jeg skrev av koder og hvorfor; hva de ulike delene av koden gjorde.

Introduksjonen varte i underkant av 25 minutter i begge klassehalvdelene. Jeg spurte elevene rett etterpå hvor forståelig de syntes gjennomgangen hadde vært på en skala fra 1 – 10. I den første halvdelen svarte elevene 5-6. I den andre halvdelen var de mer spredt i sine svar, svarene gikk fra 2 – 7 (et par sa at de godt kunne si 10 om det hjalp meg med min oppgave).

Elevene fikk vite at de skulle bygge videre på programmet mitt slik at det nå ble kastet to terninger, og summen av disse to avgjorde hvilken hest som skulle flytte. I oppgave a) på oppgavearket står det: a) Hvis vi kaster to terninger og summerer sammen antall øyne på begge terningene, hvor mange ulike summer kan vi få?

Flere av parene svarte tolv ulike summer, da etterspurte jeg hvilke tolv og elevene kom fram til at det var elleve summer. Med dette hadde de også svaret på oppgave b) Hvor mange «hester» må vi ha i hesteløpet vårt?

I oppgave c) står det at elevene skal lage programmet som simulerer hesteløpet med to terninger. Da jeg gikk rundt og observerte ble jeg overrasket over hvor greit det var for elevene å opprette en ny terning. Riktignok står det allerede i koden hvordan jeg har opprettet min, men jeg var likevel forberedt på flere spørsmål om dette. To av parene oppdaget jeg at hadde endret i variabelen «antallOyne» slik at den nå kunne få verdier fra 2 – 12. Dette var da det samme som å kaste en 11-sidet terning forklarte jeg og de endret tilbake til heltallsverdier fra 1 – 6.

Noen par var usikre på hva de skulle gjøre videre. Da spurte jeg først hva de hadde gjort til nå, og deretter hva som manglet. Med varierende grad av hint kom elevene på at de måtte kaste den andre terningen de hadde laget. Hvordan de skulle gjøre det kom det

ingen spørsmål om, de kunne se i koden hvordan jeg hadde gjort det med den første terningen.

I begge klassehalvdelene var det ikke alle par som var like godt i gang:

En gruppe startet bra, men fikk ved et uhell forlatt programmet slik at de måtte starte på nytt. Da forsvant motivasjonen og de lekte med andre p5.js programmer de kunne finne.

Et annet par kom aldri ordentlig i gang. De ville heller ikke ha hjelp.

Et tredje par ble til en gruppe på tre hvor elevene pratet mer om hverdagslige ting enn matematikken. Da skolen var på rødt smittevernsnivå og samfunnet ellers mye nedstengt ble dette mer utbredt, forståelig nok. Elevene hadde behov for sosialisering.

Hovedfokuset mitt ble etter hvert mer rettet mot de elevene som engasjerte seg i oppgaven.

Det spørsmålet som jeg fikk høre flest ganger var hvordan de skulle summere de to terningene. Jeg gjentok hva de ønsket å oppnå (summere to terninger) og spurte dem om de hadde forslag til løsning. «Kan jeg bare skrive terning1 + terning2?» var et typisk forslag elevene kom med. Jeg svarte bekreftende, men de må lagre det svaret et sted. Flere av elevene kom selv med forslaget om å opprette en ny variabel, noen kalte den «sum», andre kalte den «resultat». Hva de kalte variabelen ble påvirket av hvilke ord jeg brukte mens jeg snakket med dem: «Hvor vil du lagre resultatet/summen?».

Etter spørsmålet om hvordan de skulle summere terningene, var det ikke så mange flere spørsmål. Ved å kjøre programmet for å teste det fikk de øyeblikkelig tilbakemelding på om det fungerte slik de så for seg. De måtte legge til flere hester og sørge for at riktig hest flyttet på seg. Noen av «hestene» ble tegnet oppå hverandre, da måtte noe endres. Det var mange like endringer i koden som skulle til. Før timen var slutt hadde to grupper i første halvdel blitt helt ferdige, og en tredje gruppe nesten ferdig. I andre halvdel ble tre av gruppene ferdige og en fjerde gruppe hadde ikke blitt ferdig med å opprette alle hestene enda.

En av gruppene som fullførte ble ikke overrasket over resultatet. De sa de visste på forhånd at det ville bli sånn. Dette hadde de arbeidet mye med på barneskolen og husket det godt. De påpekte allikevel at visualiseringen var fin og nyttig.

De resterende fire gruppene som fullførte programmet visste ikke på forhånd at resultatet ville bli slik det ble. «Spyd» formen som flere kalte det var overaskende. Noen kunne forklare med en gang hvorfor det ble slik, f.eks. «Det er flere kombinasjoner som gir 7-eren, 6-eren og 8-eren». Andre trengte litt tid. Gruppene snakket sammen om programmet til slutt. Det som var tydelig var at de elevene som hadde skrevet mest på programmene kunne forklare det best. De elevene som i stor grad hadde sett på, eller ikke deltatt i det hele tatt, kunne ikke forklare «spyd-formasjonen».

Alle ble i det minste enige om at 7-er hesten var godt trent!

En elev (som ikke hadde hatt valgfag programmering) satt for seg selv i slutten av timen og programmerte videre. Kunne han få hestene til å bevege seg motsatt vei, fra høyre mot venstre? Dette kom han på helt av seg selv. Han endret også hestenes visuelle uttrykk til forskjellige «fugle-emojier» istedenfor tall.

Timene ble avsluttet med et nytt hesteløp, denne gangen med to terninger. Alle elevene gjettet på at 7-er hesten ville vinne. I andre klassehalvdel endret jeg i koden slik at hest2 og hest7 byttet plass, og stemte selv på hest2 som vinnerhesten. Jeg vant den runden.

Har elevene arbeidet som programmere?

Alle elevene har vært med på å skrive koder steg-for-steg og laget en *algoritme*. Hele hesteløpet er blitt brutt ned i deler; «først dette, så dette».

Jeg har sett flere tegn på at elevene bruker *logikk,* for eksempel når de analyserer terningkastene og kommer fram til at det er 11 ulike summer og forutser derfor at de trenger 11 hester.

Elevene har flere ganger brukt *dekomposisjon*, for eksempel når de opprettet den andre terningen. Skapelsen foregikk i flere deler: Først lage variabelen, deretter «kaste» den og til slutt utføre en beregning. Terningkastet brytes ned i mindre deler, og pusles sammen til slutt.

I den opprinnelige koden er det mange setninger som går igjen med små endringer, disse *mønstrene* har elevene utnyttet når de selv har skrevet videre på programmet. De har bl.a. funnet likhetene i koden for hvordan hestene beveger seg.

```
//Hvilken hest skal flytte?
if (terning1 == 1)
  hest1 += 1;
else if (terning1 == 2)
  hest2 += 1;
```

Figur 37: Opprinnelig kode

endret til

```
//Hvilken hest skal flytte?
if (sum == 2)
  hest2 += 1;
else if (sum == 3)
  hest3 += 1;
```

Figur 38: Etter endring

Utklippene over viser også evne til abstraksjon; elevene bryr seg ikke om alle detaljene, de fokuserer på det som er spesielt i hvert tilfelle og lar det som er generelt stå. Alle hester skal ha startposisjon og hver hest skal flyttes når en spesiell hendelse inntreffer. Den generelle koden for å flytte en hest er likt fra gang til gang, men hvilken hest som skal flytte avhenger av summen på terningene, dette er det spesielle. Elevene endrer da i grunn på to tall fra hest til hest.

Elevene har vært nødt til å gjøre *evalueringer* underveis. Hva skal de skrive? Hvor skal de skrive? Hvordan skal de skrive det? Hvorfor fungerer det ikke? Hvorfor former hestene seg som et spyd? Slike vurderinger har elevene støtt på i hele sin oppgaveløsingsprosess. Ettersom mye av koden alt var skrevet og forskjellige kodelinjer bare kunne kopieres var jeg usikker på om elevene ville forstå arbeidet de gjorde, eller om de bare kopierte. En observasjon som jeg mener peker i retning at de forstod hva de gjorde var at ingen av elevparene lagde to «antallOyne» variabler. Elevene kopierte koden for å opprette en terning, men vurderte at de ikke trengte å kopiere «antallOyne». Da har de gjort evalueringer og forstått hensikten til de ulike kodelinjene.

Å *fikle* vil si å utforske og eksperimentere. Elevene har ikke gjort så mye av dette som jeg fikk observert, men mulighetene er der. Den ene eleven som utforsket på egenhånd hvordan han kunne få hestene til å bevege seg i motsatt retning har *fiklet* med koden. Det er også mulig å undersøke hva som vil skje om noen av hestene beveger seg to steg eller flere i stedet for bare ett? Hvor mange steg må de forskjellige hestene flyttes for at det andre hesteløpet skal bli rettferdig (like stor vinnersjanse til alle hestene)?

Hesteløpet har elevene vært med på å lage selv, selv om det var jeg som i utgangspunktet hadde tatt de fleste valgene. Elevene har påvirket design av koden; hva kaller vi variabelen? Hvor skal vi skrive? De har dermed *skapt* et virtuelt hesteløp. Å endre hestenes visuelle uttrykk er også et eksempel på design.

Elevene prøvekjørte programmet flere ganger for å sjekke om det fungerte slik de hadde tenkt. Hvis «hestene» tegnet over hverandre, visste de at noe måtte endres. De oppdaget og rettet på feil, det som kalles *feilsøking*.

Det kan være fryktelig frustrerende når en oppdager at noe er feil. Elevene måtte *holde ut* for å bli ferdige. Flere ganger stoppet elevene opp fordi de hadde fått en feilkode. Konsollvinduet ble konsultert og feilen i koden ble lokalisert. Elevene måtte prøve igjen og fortsette med arbeidet.

Elevenes *samarbeid* var ikke like lett for meg å holde oversikt over i disse timene. I de to første oppleggene mine hadde én elev ansvaret for en ekstern tilkoblet datamus og den andre ansvaret for tastaturet. I tillegg hadde jeg koblet opp en større skjerm som elevene kunne arbeide på. I arbeidet med p5.js brukte elevene en bærbar datamaskin uten egen tilkoblet datamus og uten større skjerm. Da var det i stor grad kun én elev som var fysisk delaktig i kodeskrivingen, mens den andre for det meste så på. Dette kan ha vært med på å påvirke hvorfor noen ble sittende alene og skrive mens partneren ble distrahert av andre par og samtaler. Mulighetene for *samarbeid* har vært tilstede, men ble ikke utnyttet like godt av alle par. Et av parene som jeg mener samarbeidet godt vekslet på hvem som skrev og den som ikke skrev var allikevel delaktig ved å peke på skjermen.

4.3 Kjerneelementene

I denne delen av analysekapitlet vil jeg ta for meg kjerneelementene i matematikk. Jeg går tematisk til verks og tar for meg ett og ett kjerneelement. Datamaterialet jeg har samlet inn går jeg over en gang til og analyserer med hensyn på operasjonaliseringen av kjerneelementene (Tabell 5, s. 24). Jeg bruker beskrivelsene fra tabellen som sjekkpunktlister. Noen av sjekkpunktene behandler jeg individuelt, andre i bolker, ettersom noen av episodene jeg beskriver kan dekke flere sjekkpunkter.

4.3.1 Scratch – Mangekanter og vinkler

1 Utforsking og problemløsing

- **Elevene arbeider med problemer de ikke kjenner fra før**

Kari og Trine har i arbeidet sitt med mangekanter og vinkler støtt på problem de ikke kjente løsningen på fra før. Allerede med den første utfordringen, å tegne en trekant, foreslår elevene å snu katten 60°:

Kari [00:58] Så må den snu 60 grader, for hvis du skal ha en likesidet trekant så må den jo være 60.

Dette tegner ikke en trekant. Elevene visste ikke på det tidspunktet at å «snu» katten og fortsette fremover ikke var det samme som å tegne den innvendige vinkelen de ønsket.

Når elevene skulle tegne femkanten, var problemformuleringen nå kjent; de visste hva de skulle gjøre og hva de trengte av informasjon for å få det til. Men de visste ikke hvordan de skulle finne informasjonen de manglet; hvor mange grader skulle katten snu for å lage en femkant?

Kari [03:27] Er ikke det 115 eller noe? Jeg husker ikke hvor mye det er i en femkant.

Kari [03:44] Vi kan ta … Vi kan jo hvertfall prøve med … Skal vi prøve med 120 sånn bare for gøy for da øker den med 30.
Trine [03:54] Okay. Vi kan prøve … Også kanskje fem ganger. Fem.

Trine [04:18] Ja, men på … femkant … for på en firkant er det 360, så da er det … vent, jeg må tenke … (ler) nei …

Elevene vet ikke svaret, og heller ikke hvordan de skal gå fram for å finne ut av det.

Det samme skjer når elevene skal tegne syvkanten:

Kari [08:54] Eh, å ta liksom syvkant og sånn er jo bare å endre på grader greia. Så det blir jo egentlig litt kjedelig.
TC [09:01] Hvordan kan dere lage syvkanten da?
Kari [09:03] Vi må endre på, vi må gjøre sånn at han snur seg …
Trine [09:06] Endre der (peker) også i hvor mange ganger det gjentar.
Kari [09:09] Ja, mindre der, og større der.
TC [09:12] Hvor mye mindre må gradene bli da?
Kari [09:15] Nei, eh …
Trine [09:15] (ler) … Hmmm….

Elevene forstår problemet, de må endre på to tall, men vet ikke hvordan de skal komme fram til antall grader katten må snu for å tegne syvkanten.

Oppgave 2) og 3) (8.3 Oppgaveark: Scratch – Mangekanter og vinkler) hadde elevene aldri vært borti før. Å tegne mangekanter var nå kjent, men å spørre brukeren av

programmet om å oppgi svar hadde de ikke gjort før. De hadde heller ikke opprettet egne variabler før.

Elevene har gjennom hele opplegget støtt på problemer de ikke kjenner fra før.

- **Elevene diskuterer seg fram til en felles forståelse**

Trine og Kari har snakket seg gjennom hele oppgaveløsingen.

Kari [00:46] Ok. Og så … Må den jo gå sånn 100 steg kanskje.
Trine [00:50] Mhm. Kanskje det.
Kari [00:54] Skal vi skrive det? … Ja!
Trine [00:56] [???]
Kari [00:58] Så må den snu 60 grader, for hvis du skal ha en likesidet trekant så må den jo være 60.
[01:06] [???] Er det ikke bare sånn gjenta …
Trine [01:09] Tre ganger.
Kari [01:10] Mhm.

Kari bruker ordet «kanskje» som viser til en usikkerhet og er lagt fram som et spørsmål. Trine er også usikker, men de blir enige om å prøve seg med 100 steg. Kari spør så om de skal bruke «gjenta», hvor Trine skyter inn «tre ganger». Elevene har en felles forståelse av hva de gjør.

Kari [02:17] Jaa! (Hendene i været) Det ble det, det var bare det at vi glemte å slette alt.
Trine [02:24] Vi kan slette før …
Kari [02:25] Vi kan ta på sånn slett alt ting … Med den
Trine [02:29] Ja, slett alt. Før alt annet. Ja.

Elevene diskuterer når de kommer på at de ønsker å slette alt som er blitt tegnet før hver gang programmet starter på nytt. Trine foreslår at blokken «slett alt» må komme først i koden og begge forstår hvorfor de gjør det.

Da de slet med femkanten får vi også et innblikk i hvordan de diskuterer seg fram til en løsning:

Trine [04:49] Eh, jeg tror det skal være.
Kari [04:51] Det var nærme nok da. (smiler)
Trine [04:53] Kanskje det er 75?
Kari [04:56] Vent litt … Se … se … se… Det blir sånn, sånn nesten.
Trine [05:00] Ikke helt.
Kari [05:01] Det ble [???] godt nok!

Trine [05:09] 75?
Kari [05:10] Prøv 75.

Kari [05:17] Litt mye.
[05:23] 71
Trine [05:24] 73?
Kari [05:26] 72,5
Trine [05:27] Jaa … Går det an å ta …
Kari [05:29] Jeg har ikke peiling.

Trine [05:31] Desimaltall. … Nei det ble ikke noe desimaltall. Tar 73 … Rundet det opp litt.

Diskusjonen dreier seg om hvilke tallverdier de skal prøve ut og om resultatet er godt nok. Med syvkanten diskuterte de hvilke verdier de skulle prøve seg fram med, med åttekanten ble diskusjonen noe annerledes:

Kari [09:19] Så var det jo det da … En åttekant har eh …
Trine [09:25] Kan du det?
Kari [09:27] (ler) En åttekant?
Trine [09:28] Har det ikke bare dobbelt så m …
Kari [09:30] Ja, prøv å ta 45 du. Ta 45.
Trine [09:32] Okay. Ja, da er det 45 og så er det 8 ganger … For da er det … eller er det?

Med åttekanten har de en antakelse på forhånd om hvilken verdi som vil fungere. De kommer fram til at 45° er et godt forslag.

Elevene begir seg ikke inn i noen dype diskusjoner med hverandre, samtalene dreier seg i stor grad om hva de skal prøve for å komme videre. Gjennom samtalene kommer det fram at begge forstår hva de gjør og hvorfor.

- **Elevene bryter ned et problem i delproblem**

Å bryte problem ned i delproblem er noe av essensen i programmering. Algoritmer skapes nettopp som et resultat av dette.

Kari [01:31] Må den ikke gå før den snur? (snu kloss kommer etter gå)
Trine [01:34] En likesidet trekant. Eh, jo!
Kari [01:35] Også tre!

Kari [01:48] Nei, nei… Nå kan vi se at den er der. Så kommer han til å gå 100 steg og så kommer han til å vri seg 60 grader og så kommer han til å gå 100 steg, og så kommer han til å vri seg 60 grader, så kommer han til å gå 100 steg til.
Trine [01:58] Og så blir det en trekant.

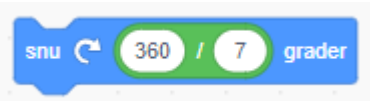
Fremgangsmåten for å tegne en trekant kan deles inn i mindre deler. Først må de få katten til å gå. Dette gjør de ved hjelp av en «gå»-blokk. Deretter må de få katten til å snu, ved hjelp av en «snu»-blokk. Dette skal så gjentas tre ganger ved hjelp av en «gjenta»-blokk. De tre delene; gå, snu og gjenta er delproblemene som er blitt løst. Når de pusles sammen blir de løsningen på hele problemet; det blir en trekant.

Da elevene fikk skapt malen sin var de fleste delproblemene løst. Det eneste de måtte endre på var nå antall gjentakelser og antall grader katten skulle snu. Antall gjentakelser var bare det samme som antall sidekanter i mangekanten. Antall grader var delproblemet som gav dem mest utfordring. Det har vi sett flere eksempler på. Et eksempel er da de oppdaget at de kunne bruke divisjon for å beregne antall grader til syvkanten:

Kari [16:38] Så det blir delt på 7 … Okay, så har vi den gamle måten. Hvor mange ganger får du plass til 7 i tre (ler)?

Kari kaller divisjon for hånd for «gamlemåten». Etter å ha løst delproblemet 360° dividert på 7 kunne denne løsningen blitt brukt som en del av løsningen på hele

problemet. Elevene brukte en operatorblokk. Det visuelle brukergrensesnittet viser at løsningen på et delproblem brukes i løsningen av hele problemet.



Figur 39: Operatorblokk inni snublokk

Figur 39 viser visuelt hvordan løsningen på beregningen i den grønne operatorblokken avgjør hvor mange grader katten skal snu.

Noe av det samme finner vi eksempler på senere også:

Kari [24:24] … liksom … For hvis vi lager en ekstra sånn variabel ting. Som heter, ehm, grader da. Ikke sant? Så sier den hvor mange grader den skal snu. Og de gradene er 36 delt på hjørner. Sånn at den snur ikke «hjørner» den snur «grader», ikke sant?

Her foreslår Kari å opprette en egen variabel som de kan kalle «grader». Antall hjørner som brukeren oppgir kan da brukes i en beregning hvor resultatet av beregningen lagres i denne variabelen. Variabelen brukes så senere i koden til å avgjøre hvor mange grader katten skal snu. Her er det flere delproblem som løses hver for seg, og som så flettes sammen til å løse et større problem. Delproblem: Be bruker om antall hjørner. Delproblem: Beregne antall grader katten må snu. Delproblem: Få katten til å gå og snu. Dette er eksempler på noen av delproblemene som først måtte løses før hele programmet kunne brukes.

- **Elevene leter etter mønster og sammenhenger**

Som beskrevet flere ganger nå, så dreide mange av oppgavene seg om å finne antall gjentakelser og antall grader. I arbeidet med dette oppdaget elevene mønster og sammenhenger. Elevene brukte det de visste fra før; alle sidekanter og innvendige vinkler er like i en regulær mangekant, i en likesidet trekant er innvendig vinkel 60° og i et kvadrat er innvendig vinkel 90°. Med utgangspunkt i dette oppdaget de nye sammenhenger:

Kari [06:21] En sekskant har dobbelt så mange kanter som en trekant, så det vil gi mening at de er dobbelt så store som en trekant.

Kari [10:06] Eh fordi, en trekant var 120 og …
Trine [10:09] Ja! Og så …
Kari [10:09] Da ble sekskanten 60 … så da … åttekant det er eh dobbelt så mange kanter som en firkant [???] alle er 90
Trine [10:15] Og da er halvparten så mange … grader.

Elevene har funnet at hvis antall sidekanter dobles, skal antall grader katten må snu halveres. Denne sammenhengen bruker de for å løse seks-, åtte- og tikanten. Men fortsatt hadde de problemer med mangekantene hvor de ikke kjente til gradene til en mangekant med halvparten så mange sidekanter. F.eks. i arbeid med syvkanten hadde de ikke en tre-og-en-halvkant å bruke som utgangspunkt.

Etter en liten dytt fra meg i form av en tabell hvor jeg systematiserte noen av funnene deres, oppdaget de et mønster:

Kari [16:13] Litt over 360, ÅÅ! Åååå jaaa … (smiler til Trine) Vi må gjøre sånn at det blir 360.

Trine	[16:17] For her er det …
Kari	[16:18] 7 delt på … 360 delt på 7.
Trine	[16:21] Her er det tre hundre [???] Her er det [???] åtti …
Kari	[16:26] Nei, men du må jo gange det med 3 … Gange det med 4, gange det med 5, gange det med 6 … Det er 360 delt på 7 … Kan vi skrive på denne?
Kari	[16:38] Så det blir delt på 7 … Okay, så har vi den gamle måten. Hvor mange ganger får du plass til 7 i tre (ler)?

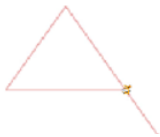
Mønsteret som oppdages er at «Antall gjentakelser» multiplisert med «Antall grader» til hver av mangekantene de hadde laget alltid ble 360°. Dermed kunne problemet med syvkanten løses ved å bruke divisjon. 360° dividert på 7 vil gi antall grader katten må snu.

- **Elevene vurderer om løsningene er gyldige**

Hver gang elevene kjører programmet sitt får de se et resultat i visningsvinduet. Dette blir det visuelle uttrykket til løsningen og elevene vurderer om det stemmer overens med intensjonene.

Figur 40 viser første gang elevene fikk tegnet en trekant. Men elevenes vurdering var at løsningen ikke var god nok da de ikke hadde slettet det de hadde tegnet gangen før.

Figur 40: Ikke godkjent



Figur 41 viser en annen versjon av trekanten. Elevene vurderte at løsningen ikke var god nok fordi de ikke hadde brukt «penn av»-blokken før de flyttet katten til sitt utgangspunkt. Dermed tegnet den et ekstra linjestykke som ikke skulle være med.

Figur 41: Ikke godkjent

I arbeid med fem- og syvkanten var det også visningsvinduet som avgjorde om elevene var fornøyd med det de hadde skrevet i algoritmen. Elevene prøvde seg fram med ulike grader og vurderte for hver gang om de virkelig hadde tegnet det de ville. Ettersom Scratch kan gi et visuelt resultat med en gang programmet kjøres, kan elevene vurdere umiddelbart om løsningen er gyldig.

Alle fem deler av dette kjerneelementet har vært en aktiv del av oppgaveløsingen.

2 Modellering og anvendelser

- **Elevene lager modeller som beskriver noe fra virkeligheten**

Programmet som elevene har laget beskriver hvordan en virtuell katt kan tegne ulike mangekanter ved å bruke sin egen kropp. Dette kan ses på som en modell for hvordan et menneske (eller annen fysisk enhet) fysisk kan lage mangekanter med kroppen sin. Programmet kan beskrive noe fra virkeligheten, men jeg ville allikevel ikke kalt dette en

matematisk modell basert på noe fra virkeligheten. Med virkelighet, menes her dagliglivet, arbeidslivet og samfunnet ellers. Når i hverdagen får du bruk for å bevege deg langs omkretsen til en regulær mangekant? Men når det er sagt så inneholder dagliglivet vårt stadig flere teknologiske oppfinnelser som skal gjøre livet vårt enklere. Robotgressklippere og -støvsugere blir stadig mer vanlige. Bevegelsesmønstrene til disse enhetene må programmeres, så kanskje mangekantprogrammet ikke er så langt fra virkeligheten likevel?

- **Elevene bruker sin matematiske kunnskap i nye situasjoner**

I arbeid med trekanten kjente elevene allerede til at innvendig vinkel i en likesidet trekant er 60° og at vinkelen til en rett linje er 180°. Denne kombinasjonen av kunnskap gjorde at de raskt fant ut av at katten måtte snu seg 120° ($180° - 60°$) for å tegne trekanten. Elevene har brukt sin matematiske kompetanse om geometri i en ny situasjon hvor de skulle programmere en katt.

Å tegne en sekskant var også en ny situasjon for elevene. Her brukte de igjen kunnskap de alt hadde til å løse oppgaven. Sekskanten har dobbelt så mange sidekanter som trekanten, da gav det mening for elevene at den innvendige vinkelen også burde være dobbelt så stor og dermed at den utvendige vinkelen måtte halveres i forhold til trekanten.

Da elevene oppdaget mønsteret at «Antall gjentakelser» multiplisert med «Antall grader» alltid ble 360°, anvendte Kari kunnskapen sin om at multiplikasjon og divisjon er motsatte regnearter og fant fort ut av at for å finne antall grader katten måtte snu for å tegne syvkanten kunne de ta 360° dividert på 7.

De fleste oppgavene denne timen var nye for elevene (da programmering i seg selv var nytt) og den matematiske kunnskapen de har brukt har dermed blitt brukt i nye situasjoner.

3 Resonnering og argumentasjon

De tre kjennetegnene på resonnering og argumentasjon i Tabell 5: Operasjonalisering av kjerneelementene glir inn i hverandre, og jeg omtaler dem derfor samlet i denne delen av analysen.

- **Elevene utformer egne resonnement**
- **Elevene forstår andres resonnement**
- **Elevene begrunner fremgangsmåter**

Å resonnere vil si å forklare hvordan man tenker og begrunne løsningene sine. Det handler om å følge, vurdere og forstå matematiske tankerekker. Det har vi mange eksempler på i elevenes oppgaveløsing.

Kari [01:48] Nei, nei… Nå kan vi se at den er der. Så kommer han til å gå 100 steg og så kommer han til å vri seg 60 grader og så kommer han til å gå 100 steg, og så kommer han til å vri seg 60 grader, så kommer han til å gå 100 steg til.

Trine [01:58] Og så blir det en trekant.

Kari [02:00] Ja, men det kan også være at den ikke vrir seg innover, at han bare blir sånn (viser med fingeren at det ikke blir en trekant).

Kari forklarer algoritmen elevene har skrevet; katten skal gå 100 steg og snu seg 60°, tre ganger. Trine forstår Karis resonnement og vurderer at resultatet vil bli en trekant. Samtidig er Kari forberedt på at det muligens ikke blir en trekant, men starten på noe annet (som viste seg å være en sekskant). Hun argumenterer for at katten kanskje ikke «vrir seg innover». Elevene følger en rekke matematiske instrukser, vurderer gyldigheten og forstår utfallet. Det viste seg at de måtte få katten til å snu 120° for å tegne en trekant.

Kari [04:31] Fordi, når vi tok 60 i stad så ble det jo liksom starten på (viser med finger) et eller annet. Så hvis vi prøver å ta sånn rundt fem … ti, nei ikke femti, eh … Prøv å skriv 70.
Trine [04:43] 70?
Kari [04:44] Ja, det er 10 mer enn 60 … Det er matte.

I arbeidet med femkanten resonnerer Kari seg fram til at et godt sted å starte er 70°. 60° kunne det ikke være, 50° var enda mindre, så de økte heller med 10° og forsøkte seg med 70°. Etter en del prøving og feiling lykkes elevene med 72°.

Etter å ha lykkes med femkanten sa Kari «nå har vi lært det» uten å spesifisere hva de hadde lært. Dette spurte jeg om da jeg fikk sjansen.

TC [10:37] Det jeg lurer på … Hva har dere lært? Når det stod 72 grader?
Kari [10:41] Nei, vi har lært at eh…
Trine [10:42] Vi har lært at det …
Kari [10:43] Det er 108 grader i en regulær femkant … ikke sant? … Det blir jo det? For 108 + 72 er 180.

Elevene kan resonnere seg fram til at den innvendige vinkelen i en femkant er 108° siden katten måtte snu 72° for å lage femkanten og summen av 108 og 72 er 180.

I arbeid med sekskanten foreslår Kari å prøve 60°.

Kari [06:21] «Jeg føler det i meg». Ja, fordi. Ja! Fordi, ehm … En sekskant har dobbelt så mange kanter som en trekant, så det vil gi mening at de er dobbelt så store som en trekant.
Trine [06:29] Okay, kanskje.

I utsagnet over argumenterer Kari for gyldigheten av sitt forslag. «En sekskant har dobbelt så mange kanter som en trekant», da bør også de innvendige vinklene være dobbelt så store og dermed må katten snu seg 60° for å tegne en sekskant. Trine forstår resonnementet, men er ikke helt overbevist om resultatet. Det viser seg at det fungerte, og elevene har funnet en fremgangsmåte de forstår og som gir mening for dem.

Kari [09:03] Vi må endre på, vi må gjøre sånn at han snur seg …
Trine [09:06] Endre der (peker) også i hvor mange ganger det gjentar.
Kari [09:09] Ja, mindre der, og større der.

I arbeid med syvkanten argumenterer elevene for hvilke to tall som må endres i malen sin. Flere hjørner i mangekanten krever at «gå og snu» gjentas flere ganger, og antall grader katten snur må bli mindre.

Kari er i stor grad den som utformer resonnementene underveis, men Trine viser at hun forstår resonnementene og kan bruke dem selv.

Kari	[10:06] Eh fordi, en trekant var 120 og …
Trine	[10:09] Ja! Og så …
Kari	[10:09] Da ble sekskanten 60 … så da … åttekant det er eh dobbelt så mange kanter som en firkant [???] alle er 90
Trine	[10:15] Og da er halvparten så mange … grader.
Kari	[10:18] Mhm. Halvparten av 90. … Så det går ikke an å lage en 16-kant, for da må det være komma (elevene trodde ikke at Scratch godtok desimaltall).

Her forklarer elevene hvordan de fant ut at åttekanten krevde at katten skulle snu seg 45°. De sammenligner trekanten og sekskanten, bruker dette som grunnlag, og benytter seg av det samme mønsteret for å sammenligne åttekanten med firkanten. Ettersom åttekanten har dobbelt så mange kanter; vil vinkelen halveres fra 90° til 45°.

Elevene har i løpet av denne timen utformet egne resonnement, forstått den andres resonnement og begrunnet sine fremgangsmåter.

4 Representasjon og kommunikasjon

- **Elevene representerer matematiske begrep, sammenhenger og problemer**
- **Elevene bruker matematisk språk**
- **Elevene veksler mellom ulike representasjoner**

Representasjoner er måter å uttrykke matematiske begrep, sammenhenger og problemer på. I arbeidet med Scratch og mangekanter har elevene representert mangekanter på forskjellige måter.

100 er et tall, her vist som tallsymbol. Elevene har i arbeidet sitt uttrykt «hundre» verbalt, skrevet det med tallsymbol i algoritmen sin og visuelt sett hva lengden 100 steg ser ut som i visningsvinduet. De veksler da mellom ulike representasjoner.

Elevene har løst oppgaver hvor de har skrevet algoritmer for å tegne regulære mangekanter. Mangekantene representeres da visuelt i visningsvinduet og som et sett skriftlige instrukser i skriptvinduet. Elevene har jevnlig vekslet mellom disse representasjonene: De sier det verbale navnet, for eksempel «sekskant», skriver «6 gjentakelser og 60°», og ser det visuelle resultatet.

Den matematiske sammenhengen som elevene oppdaget i slutten av timen, at katten må snu 360° dividert på antall hjørner for å lage en hvilken som helst regulær mangekant, representeres som en algoritme. I arbeidet med å komme fram til denne algoritmen har elevene brukt et matematisk språk. De har beskrevet lengder og størrelser på vinkler, snakket om dobling og halvering, beskrevet koordinater ved hjelp av x- og y-verdier og brukt regneoperasjoner.

Programmeringsblokkene er i seg selv representasjoner. Gjenta-blokken for eksempel representerer programmeringsprinsippet «løkke». Koden inni denne blokken vil gjentas et visst antall ganger før programmet går videre til neste steg. Dette er en representasjon av et matematisk programmeringsbegrep.

Figur 42: Gjenta-blokken

I dette arbeidet med Scratch har elevene brukt visuelle, verbale og symbolske representasjoner. De har vekslet mellom disse representasjonene og brukt et matematisk språk i prosessen med å utarbeide representasjonene.

5 Abstraksjon og generalisering

- **Elevene formaliserer tanker, strategier og/eller matematisk språk**
- **Elevene oppdager sammenhenger og strukturer som de formaliserer ved hjelp av algebra og/eller andre hensiktsmessige representasjoner**

Allerede i sin første mangekantmal har elevene vært innom kjerneelementet abstraksjon og generalisering. De har trukket ut og generalisert de essensielle egenskapene til en *n*-kant; *n* antall sidekanter, *n* antall hjørner og samme vinkel i alle hjørner. Denne strukturen har de formalisert i en algoritme hvor de kun endrer på to tall for å tegne forskjellige mangekanter.

En annen sammenheng elevene oppdaget var «dobling og halvering» strategien. Hvis antall hjørner i mangekanten dobles, må vinkelen katten snur halveres. Denne sammenhengen formaliserte de til en strategi som de kunne benytte for å tegne flere mangekanter. Hadde de hatt en tre-og-en-halvkant, kunne de brukt denne til å tegne syvkanten.

I siste oppgave lagde elevene et program som ba brukeren oppgi antall hjørner, så ville programmet tegne en regulær mangekant med det gitte antall hjørner. Programmet er et godt eksempel på abstraksjon og generalisering. Elevene har oppdaget sammenhenger og strukturer som de formaliserer. Ønsker brukeren å tegne en regulær *n*-kant, må katten snu 360° dividert på *n*. Dette siste programmet inneholder alle kjennetegnene på abstraksjon slik Wing (2011) beskriver det (s. 14).

6 Matematiske kunnskapsområder

- **Elevene arbeider med fagstoff fra de matematiske kunnskapsområdene**

I beskrivelsen av kjerneelementet *matematiske kunnskapsområder*, står det: «Algebra handlar om å utforske strukturar, mønster og relasjonar og er ein viktig føresetnad for at elevane skal kunne generalisere og modellere i matematikk. […] Geometri er viktig for at elevane skal utvikle ei god romforståing.» (Utdanningsdirektoratet, 2020b)

I arbeidet sitt med Scratch har elevene arbeidet med algebra og geometri. De har utforsket strukturer, mønster og relasjoner og generalisert sine funn i form av algoritmer. De har arbeidet med variabler; opprettet, tilordnet og brukt dem i algoritmene sine. Utgangspunktet for dette arbeidet har vært mangekanter og vinkler; viktige geometriske begrep som tilhører de matematiske kunnskapsområdene.

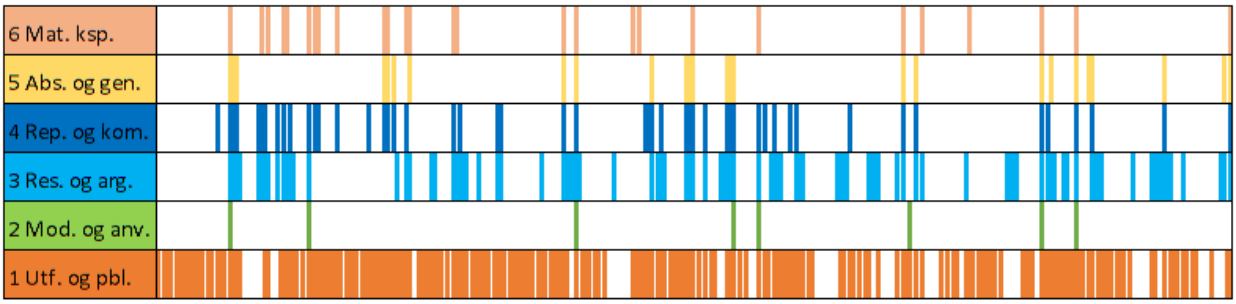


Diagram 1: Kjerneelementene, mangekanter og vinkler

Diagram 1 viser visuelt hvordan utsagnene til elevene under oppgaveintervjuet kan knyttes til kjerneelementene. Diagrammet er kronologisk; første utsagn helt til venstre og siste utsagn helt til høyre. Alle utsagn er med i oversikten, selv mine egne, og derfor er det noen utsagn uten fargekode.

4.3.2 Scratch – Lineære funksjoner

1 Utforsking og problemløsing

- **Elevene arbeider med problemer de ikke kjenner fra før**

De utvalgte elevene til dette opplegget hadde tidligere erfaring med Scratch som programmeringsspråk og derfor kan ikke programmering sies å være nytt for akkurat disse. De første oppgavene var heller ikke helt ukjente da vi hadde gjennomgått lignende oppgaver i introduksjonen som elevene hadde tilgang til på oppgavearket. Men elevene støtte likevel på problemer:

Erik	[05:50] Om ti år er du 1994
(Latter!)	
David	[05:53] Nei, nei , nei…

Elevene skulle lage et program som regnte ut alderen til en person om 10 år. Programmet spurte brukeren om alderen sin nå, elevene skrev inn 15, og resultatet ble 1994. Dermed måtte elevene arbeide med et problem de ikke kjente fra før; hvorfor ble det feil?

Videre støtte elevene ikke på noen problemer i hverken oppgave b), c) eller d). De hadde riktignok ikke arbeidet med disse oppgavetypene før, men de kjente så godt til programmeringsspråket at det meste ble løst på første forsøk.

De virkelige problemene oppstod derimot da vi også brukte GeoGebra og jeg stilte muntlige spørsmål til elevene. Kunne de lage en funksjonsmaskin hvor grafen ikke ble en rett linje? Først svarte Erik at det ville være umulig, men så kom David på idéen om å bruke en variabel med et tilfeldig tilordnet tall i beregningen.

David	[31:48] Vi kunne jo se at her er det sånn 2 tall (peker på dividert på 2 i koden) også kunne det ha vært sånn tilfeldig til hvilken det byttet…
Erik	[31:55] Det kunne variert på den…
David	[31:59] Men da er det tilfeldig, da er det ikke noe sånn rytme, eller.

Elevene har da utviklet en metode for å løse et problem de ikke kjente fra før.

Etterpå ba jeg dem om å lage en funksjonsmaskin hvor den tilhørende grafen ville være parallell med en av de andre grafene de hadde tegnet. Her også støtte de på problemer. De klarte å tegne en parallell graf i GeoGebra, men slet med å finne uttrykket til den grafen.

For disse to elevene var det ikke programmeringsspråket som var utfordringen. Det var derfor helst mot slutten av timen, da jeg gav dem oppgaver muntlig, at de arbeidet med problemer de ikke kjente fra før.

- **Elevene diskuterer seg fram til en felles forståelse**

Ettersom begge elevene var godt kjent med Scratch, var det i starten ikke så mye diskusjon, men mer enighet om hva som måtte gjøres. De delte tankene sine med hverandre og viste en felles forståelse.

Da de oppdaget hva feilen var i sitt første program diskuterte de hvordan de skulle løse det.

David	[08:25] Skal vi forandre på verdi… variabelen
Erik	[08:29] Vi tar heller bare «Hvilket år er du født?» isteden.
David	[08:32] Hehe, så vi bare avoider problemet. …. Eh, stor «h».

David foreslo å endre på variabelen, mens Erik heller ville endre på spørsmålet. David forstod hva utfallet ville være; de unngikk problemet med utregningen sin og endret forutsetningen for oppgaven.

Da de i oppgave b) skulle sette sammen fornavnet og etternavnet til brukeren av programmet ønsket de mellomrom mellom de to variablene.

Erik	[12:41] Sett sammen fornavn … Nå tester vi bare, hvis du bare skriver
David	[13:00] Oi, mellomrom, shiii. … Eh, ehm. Må vi lage, nei vi må ikke…
Erik	[13:07] Hvis du bare tar mellomrom… vent litt… det må da…
David	[13:11] Men kanskje hvis du bare kaller det for, for, nei… bare glem det.
Erik	[13:16] Nei, hvis du bare skriver… det funker da, men hvis vi bare, du bare skriver mellomrom…
David	[13:20] Å ja, jajaja. (skriver fornavnet med mellomrom på slutten)

De diskuterte hvordan de skulle løse problemet og kom fram til at brukeren selv må trykke mellomrom på slutten av fornavnet sitt.

I oppgave d) da de skulle lage nye funksjonsmaskiner diskuterer de også hvordan de skal løse problemet.

Erik	[20:18] Eh, skal vi gå inn og endre?
David	[20:31] Skal vi legge til noe minus? Sånn at det blir gange 5 pluss 3 og så minus… 100 eller noe sånt?
Erik	[20:37] Vi kan også bare ta, i stedet for
David	[20:39] Ja, ta delt på. I stedet for gange.
Erik	[20:43] Da risikerer vi at det blir kommatall, men det får gå. Jeg legger bare den der midlertidig (flytter multiplikasjonsklossen), operatør, det er deling… Svaret delt på…

David [21:01] To, skal vi ta to? Da blir det enkelt.

Elevene blir enige om å endre beregningen fra multiplikasjon til divisjon og begge viser forståelse for både de individuelle blokkene og sammensetningen av algoritmen.

- **Elevene leter etter mønster og sammenhenger**

I introduksjonen til timen blir elevene bedt om nettopp det å lete etter mønster og sammenhenger. Ett tall går inn i funksjonsmaskinen, det foretas en utregning etter et bestemt uttrykk, ett annet tall kommer ut av maskinen. Elevene skal sende flere tall inn i maskinen slik at de får mer data og kan bruke dette til å komme fram til funksjonsuttrykket.

Et av målene med undervisningsopplegget er at elevene skal lage sin egen funksjonsmaskin. Programmeringsoppgavene i Scratch er laget for nettopp dette formålet. Med flere forskjellige funksjonsmaskiner kan elevene lete etter sammenhenger mellom disse og oppdage at de (sannsynligvis) alle kan representeres med rette grafer. Ettersom elevene satt alene med meg på et grupperom, adskilt fra klassen, lagde de flere funksjonsmaskiner selv og lette etter sammenhenger mellom disse.

Allerede ved plassering av de første punktene i koordinatsystemet la Erik merke til noe:

Erik [23:43] Jeg tror jeg vet hva som kommer til å skje

Erik [23:54] At vi fortsetter med samme mønster oppover. Istedenfor å bruke den maskinen der, kunne du bare sett at du skal en bort og to opp, og da ville … svaret… du ville hatt 6 og fått ut 13 (peker på koordinatsystemet). Så da vil du kunne brukt GeoGebra eller koordinatsystemet til å finne ut hva svaret ville vært før du hadde tastet inn i maskinen. … Det er vel et verktøy, hvis du for eksempel hadde tatt den linja så ville den gått oppover der…

Erik har oppdaget et mønster i måten punktene funksjonsmaskinen gir plasseres i koordinatsystemet. Han sier at du skal «en bort og to opp», og sikter til stigningen til grafen som da ble en rett linje.

Senere da elevene skulle lage en funksjonsmaskin som hadde en graf parallell til en annen de alt hadde tegnet inn, gikk de også på leting etter mønster.

Erik [35:25] Ja, parallell. Da måtte 10 bli til 10, 14 bli til 12, har vi noe mønster ut av det? … 6 til 8… På et eller annet tidspunkt vil det bytte retning, så jeg tror ikke det er helt mulig men.

David [35:44] Vi må bare finne forholdet til de to tallene.

Erik [35:48] Problemet er at det varierer, for du ser at 2 blir til 6, 4 blir til … nei, det blir desimaltall, må zoome ut da. … 2 blir 6, 3 blir 6,5 ca. , 4 blir til 7, 6 blir til 8, 8 blir til 9, 10 blir 10, 12 blir til 11, nei vent litt… pluss fire, pluss 3, pluss to, pluss 1, pluss 0… Der blir det minus, så det ville byttet, 10 er midtpunktet. Det ville byttet fra pluss til minus, så det ville vært litt vanskelig, men det kunne kanskje vært mulig, men det tror jeg ikke vi får til akkurat nå.

Erik har oppdaget et mønster, men klarer ikke å bruke det til å finne uttrykket som kan brukes i funksjonsmaskinen.

Med få minutter igjen av timen utfordret jeg elevene kjapt på om de kunne se på forhånd at to uttrykk gir parallelle linjer. Elevene svarte fort at to av uttrykkene var parallelle fordi begge hadde «ganger 2» skrevet i funksjonsmaskinen.

David	[40:11] Begge er ganger to.
Erik	[40:17] Bare der har vi tatt min…, vi har tatt bort 1, så da blir det flyttet et hakk ned. Egentlig så ville linjen gått sånn, men siden vi flyttet den et hakk ned vil den forskyve seg bortover her.
TC	[40:30] Så du mener at du kan lage enda en parallell linje?
David	[40:32] Ja.
Erik	[40:32] Mhm. Da kan vi ta, i stedet for, pluss 1 da tar vi pluss 2.
David	[40:37] Mhm.

Elevene har oppdaget en sammenheng mellom uttrykk og graf.

- **Elevene bryter ned et problem i delproblem**

I arbeidet med å utforme algoritmer er det å bryte ned problem i delproblem helt naturlig. Alle blokkene utfører hver sin del av hele algoritmen og løser da et delproblem av hele problemet.

Et eksempel på dette som ikke er knyttet til programmeringsspråkets blokker fikk vi se da elevene skulle lage en funksjonsmaskin som hadde en graf parallell til en av de andre de alt hadde tegnet i koordinatsystemet. For å løse dette problemet kan en beskrive stegene elevene tok slik:

1) Lage en parallell linje i GeoGebra
2) Finne uttrykket til denne linjen
3) Bruke uttrykket til å lage en funksjonsmaskin

Første steg løste de i GeoGebra ved å velge en av linjene de alt hadde, deretter så de for seg en linje parallell til denne. De så etter to definerte punkt som de kunne trekke en linje gjennom. Punktene ble plassert og linjen trukket. Andre steg var å lete etter mønster. Som forklart tidligere, slet elevene med å finne mønsteret første gang. De tegnet derfor en ny linje parallell til en annen graf og så at 1 -> 2, 2 -> 4, 3 -> 6, altså $y = 2x$. Dette brukte de som uttrykk i funksjonsmaskinen; «sett output til svar ganger 2 pluss 0». Og da var problemet løst.

- **Elevene vurderer om løsningene er gyldige**

Når programmet kjøres i Scratch får brukeren øyeblikkelig svar på om det funger som tenkt. Elevenes første program skulle regne ut alderen til brukeren om 10 år, men de oppdaget med en gang at løsningen deres ikke var gyldig. En 15 år gammel bruker skulle ikke bli 1994 år gammel om 10 år.

Et program som skal si for- og etternavnet til en person kan heller ikke svare «olenormann» uten mellomrom. Elevene vurderer selv at løsningen deres ikke er god nok og kommer med forslag til forbedringer.

Erik forsøkte seg på å forutse verdier fra funksjonsmaskinen ved hjelp av koordinatsystemet.

Erik [26:12] Det krysningspunktet der. Så ville vi f.eks. kunne tatt 11 in og funnet ut at det blir 54 uten å trenge å bruke maskinen.

Ved å bruke funksjonsmaskinen fant de ut at 11 inn gav 58 ut. Eriks løsning var ikke helt gyldig. Noe han også gav uttrykk for før funksjonsmaskinen ble brukt.

Erik [26:22] Eller det kan være at linja blir litt annerledes… Jeg kan trekke en linje så blir det litt lettere å se, for det blir sannsynligvis litt feil. … Eller mest sannsynlig bli et annet tall.

Ettersom elevene alltid hadde en tydelig forventning i forhold til hva som ville skje i visningsvinduet eller GeoGebra, kunne de hele tiden selv vurdere om løsningene deres var gyldige.

2 Modellering og anvendelser

- **Elevene lager modeller som beskriver noe fra virkeligheten**

Funksjoner er veldig nyttige for å lage modeller fra noe i virkeligheten, men i denne timen har elevene kun arbeidet med uttrykkene og grafene uten å knytte dem til noe konkret. Dermed kan jeg ikke si at elevene har arbeidet med modellering fra virkeligheten.

- **Elevene bruker sin matematiske kunnskap i nye situasjoner**

Jeg tolker «nye situasjoner» litt på samme måte som «nye problemer». Elevene har brukt kunnskap om beregninger til å løse delproblem av algoritmene sine, men siden elevene var så vant til Scratch fra før kan jeg ikke si at dette var nye situasjoner.

Å forutse punkt i et koordinatsystem basert på verdier fra en funksjonsmaskin vil jeg derimot se på som en ny situasjon. Og motsatt, å bruke linjen til å forutse hva funksjonsmaskinen vil svare, er også en ny situasjon.

Erik [28:23] Kan ta, hvis jeg kan finne en. (zoomer og drar i grafikkfeltet.) Der! … 106 ville blitt til 56. … Og det kan vi teste med maskinen hvis vi skriver inn 106. (endrer vindu til Scratch, tester)… Jepp, det ble riktig.

Erik har oppdaget at funksjonsmaskinen og grafen representerer det samme og kan derfor bruke den kunnskapen til å forutse verdier. Hvis grafen går gjennom punktet (106 , 56), vil det bety at hvis 106 går inn i funksjonsmaskinen, er det 56 som kommer ut.

3 Resonnering og argumentasjon

- **Elevene utformer egne resonnement**
- **Elevene forstår andres resonnement**
- **Elevene begrunner fremgangsmåter**

Med dette datamaterialet er det ikke alltid lett å få tak i elevenes resonnement og begrunnelser ved å kun se transkripsjonen. Det er mye peking og bevegelser på

skjermen som er deler av argumentasjonen. I tillegg går elevene raskt i gang og raser seg gjennom de første oppgavene.

Erik styrte datamusen og David tastaturet, da ble det ofte Erik som snakket og puslet blokkene mens David skrev.

Erik [12:09] Fint, da har vi det. Sett etternavn til 0. Koble sammen de to. Spør først, gi svaret, hva er etternavnet ditt? Da må vi ta å sette etternavnet til det som vi får som svar. (Gestikulerer med fingeren, først dette, så dette…) Så må han si… Nå kan vi i hvert fall ikke rote til med de tallene. Skal vi se, «sett sammen». Da må vi sette sammen…

David [12:40] på variabler

Erik prater seg gjennom hva de gjør, peker og plasserer blokkene i ønsket rekkefølge. David forstår det Erik sier og skyter inn hvor i Scratch Erik kan finne blokkene han er på jakt etter. Erik vil sette sammen fornavn og etternavn. Variablene «fornavn» og «etternavn» finnes i kategorien *variabler*. Dette er et eksempel på Erik som utformer et resonnement; vi må spørre først, deretter tilordne variabelen «etternavn» svaret, så må katten si både fornavnet og etternavnet, dette må vi sette sammen ved hjelp av «sett sammen» blokken … og David som forstår resonnementet.

Da jeg spurte om det var mulig å lage en linje med en funksjonsmaskin som ikke ville ha en rett linje som graf resonnerte David seg fram til at en kunne bruke et tilfeldig tall i koden, noe som varierte fra gang til gang.

David [31:48] Vi kunne jo se at her er det sånn 2 tall (peker på dividert på 2 i koden) også kunne det ha vært sånn tilfeldig til hvilken det byttet…

Erik [31:55] Det kunne variert på den…

David [31:59] Men da er det tilfeldig, da er det ikke noe sånn rytme, eller.

Men med den løsningen ville det ikke vært et mønster å finne; ingen rytme. David forklarer hvordan han tenker og begrunner, Erik forstår resonnementet.

Da elevene stod fast med å finne uttrykket til den første parallelle linjen de lagde (Tabell 8: Erik fant mønster) forklarer Erik hvilket mønster han har funnet og at det vil være vanskelig å finne uttrykket til linjen. Dette begrunner han med at differansene endrer fortegn, hvor 10 er «midtpunktet».

Til slutt i timen oppdaget elevene at det var mulig å lage parallelle linjer ved å beholde «ganger 2» men endre på hva som ble lagt til eller trukket fra.

Erik [42:43] Så vi beholder den der, ja, ganging og deling, men vi forandrer på… Ganging eller deling, samme tall blir valgt, men vi forandrer på pluss og minus, Og når det er pluss, ja, det går oppover (linja forskyves oppover), eller om det går nedover, eller, tallet som blir plusset på. Så vi kunne, der ser du at det er minus, vi kunne for eksempel tatt pluss 5, da ville den ha flyttet seg 5 bortover sånn.

Erik bruker også hendene sine når han begrunner hvordan linjene vil forskyves i koordinatsystemet avhengig av hva de plusser på. Når han sier «bortover», viser han oppover på y-aksen.

4 Representasjon og kommunikasjon

- **Elevene representerer matematiske begrep, sammenhenger og problemer**
- **Elevene bruker matematisk språk**
- **Elevene veksler mellom ulike representasjoner**

Elevene har i dette undervisningsopplegget arbeidet med ulike representasjoner av lineære funksjoner. Først ble de introdusert til funksjonsmaskinen som representerer et funksjonsuttrykk. Deretter skulle elevene lage egne funksjonsmaskiner, da måtte de endre på uttrykket. Til slutt ble tallparene, innverdiene og utverdiene, til de forskjellige maskinene plottet i et koordinatsystem og elevene oppdaget at de produserte rette linjer. Funksjonene kan da representeres med en maskin, med et uttrykk, eller med en graf. I tillegg tegnet lærer en tabell på tavlen over inn- og utverdiene (denne tegnet jeg på ark for elevene på grupperommet) slik at funksjonen også er blitt representert ved hjelp av en verditabell. Elevene har også representert funksjonen med ord, «ganger det med 5 og så legger til 3».

Stigningen til en graf formulerte Erik på eget initiativ som «en bort, to opp». Da representerte han stigningen ved hjelp av ord. Senere fant elevene ut av at to linjer var parallelle hvis de begge hadde innverdier som ble «ganget med 2». At disse to oppdagelsene begge representerte det samme; stigningen til grafen, ble det ikke tid til å gå nærmere inn på.

Det jeg bet meg mest merke i underveis i intervjuet var hvor naturlig det var for elevene å veksle mellom Scratch sin representasjon og GeoGebra sin representasjon. De klikket seg inn og ut av de forskjellige programmene uten å bli forvirret eller overrumplet av det de holdt på med. De testet innverdier i Scratch, og kontrollerte løsningen i GeoGebra. Og motsatt, fant koordinater i GeoGebra, og testet dem i funksjonsmaskinen.

I denne timen har elevene utforsket forskjellige representasjoner av lineære funksjoner.

5 Abstraksjon og generalisering

- **Elevene formaliserer tanker, strategier og/eller matematisk språk**
- **Elevene oppdager sammenhenger og strukturer som de formaliserer ved hjelp av algebra og/eller andre hensiktsmessige representasjoner**

Abstraksjon er kjernen i programmering. Elevene har i arbeidet sitt med å utforme algoritmer brukt variabler som tar vare på «det spesielle» i hvert tilfelle mens resten av koden utgjør «det generelle».

Graden av abstraksjon økte da elevene begynte å tegne punkter i GeoGebra. Erik forklarte at den rette linjen representerte alle verdiene funksjonsmaskinene kunne produsere. Det var ikke nødvendig å taste inn nye verdier i funksjonsmaskinen, vi kunne «forutsi hva maskinen ville svare». Funksjonsmaskinen beskriver en sammenheng som kan representeres som en linje i et koordinatsystem. Gjennom denne oppdagelsen kan de også gå motsatt vei. Opprette linjer som viser en lineær sammenheng mellom x- og y-aksen og oversette dette til et algebraisk uttrykk.

Et annet eksempel på abstraksjon og generalisering er da elevene forklarte hvordan de kunne lage nye parallelle linjer. Ved å beholde «ganger 2» men endre hva som ble lagt

til, kunne de «forskyve» linjen opp og ned y-aksen. Her begrunner de ut i fra et algebraisk uttrykk hvordan den grafiske fremstillingen vil bli.

6 Matematiske kunnskapsområder

- **Elevene arbeider med fagstoff fra de matematiske kunnskapsområdene.**

I beskrivelsen av kjerneelementet står det: «Algebra handlar om å utforske strukturar, mønster og relasjonar og er ein viktig føresetnad for at elevane skal kunne generalisere og modellere i matematikk. Funksjonar gir elevane eit viktig verktøy for å studere og modellere endring og utvikling.» (Utdanningsdirektoratet, 2020b)

I arbeid med oppgavene i dette undervisningsopplegget har elevene utforsket lineære strukturer, for eksempel da Erik oppdaget strukturen «en bort, to opp». De har oppdaget mønster i hvordan funksjonsmaskinene behandler innverdiene for å produsere utverdiene. Disse mønstrene har de skrevet som algebraiske uttrykk. Elevene har også oppdaget relasjoner mellom inn- og utverdiene og sett at disse kan brukes som x- og y-verdiene i et koordinatsystem. Elevene studerte relasjonen mellom x- og y-verdiene til en linje da de forsøkte å finne et algebraisk uttrykk til den linjen.

Elevene har arbeidet med variabler, skrevet uttrykk og tegnet grafer. De har arbeidet med algebra og fått en introduksjon til funksjoner.

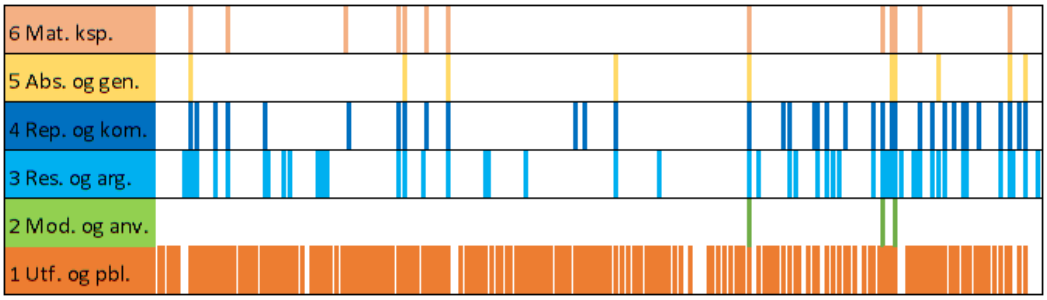


Diagram 2: Kjerneelementene, lineære funksjoner

Diagram 2 viser hvordan elevenes arbeid med dette opplegget har vært innom kjerneelementene. Alle utsagnene i løpet av intervjuet har som forklart tidligere fått en fargekode og er her presentert i kronologisk rekkefølge, fra venstre mot høyre.

4.3.3 p5.js – Sannsynlighet

Ettersom jeg ikke har innhentet elevarbeid til dette opplegget vil kommentarene i dette delkapitlet i stor grad være basert på mine egne observasjoner etter å ha gjennomført opplegget i to halve klasser. Jeg har plukket ut episoder som jeg husker fra forskjellige elevpar og ikke alle beskrivelser vil gjelde alle elevene.

1 Utforsking og problemløsing

- **Elevene arbeider med problemer de ikke kjenner fra før**

Å simulere et hesteløp med to terninger var et problem elevene ikke hadde arbeidet med før. Programmeringsspråket p5.js var også ukjent for de fleste (tre av elevene hadde arbeidet med det før, tre av elevene hadde arbeidet med et lignende språk før). Gjennom dette arbeidet har elevene laget et program; en algoritme; en metode, som løser et problem de ikke kjente fra før.

- **Elevene bryter ned et problem i delproblem**

For å løse problemet måtte elevene bryte problemet ned i delproblem. Først opprette en ny terning, så kaste terningen, så summere terningene og til slutt flytte riktig hest. Denne måten å arbeide på er helt sentral i programmering.

- **Elevene leter etter mønster og sammenhenger**

Elevene har lett etter mønster og sammenhenger i kodelinjene og i terningkastene. I arbeidet med å finne ut av hvor mange ulike summer en kan få med kast av to terninger, løste elevene oppgaven på forskjellige vis. Noen begynte å liste opp alle mulige kombinasjoner og oppdaget f.eks. at summen 2 og 12 kun var mulig hvis terningene begge viste en 1-er eller en 6-er. Her er det mulig at noen oppdaget at det var flest kombinasjoner som gav summen 7.

- **Elevene vurderer om løsningene er gyldige**

En gyldig løsning av problemet vil være en visuell representasjon av et hesteløp med 11 hester. Etter at elevene hadde opprettet to terninger, kastet og summert dem, var det flere som kjørte programmet og så at det fortsatt kun var seks hester. De vurderte da at de ikke var kommet i mål, og gikk gjennom koden sin på nytt for å finne hva mer de måtte endre på.

- **Elevene diskuterer seg fram til en felles forståelse**

Diskusjon blant elevene underveis var vanskelig å få med meg ettersom jeg stadig gikk fra par til par. Men på slutten ble de alle bedt om å forklare hvorfor 7-er hesten alltid vant. Da snakket i hvert fall alle parene med hverandre uten at jeg har god oversikt over hva de sa.

2 Modellering og anvendelser

- **Elevene lager modeller som beskriver noe fra virkeligheten**

Simuleringen av terningkast er en modell som beskriver noe fra virkeligheten. Vi kunne kastet to terninger flere tusen ganger, men det er tidsbesparende, lett å gjenta og enkelt å gjøre justeringer hvis en istedenfor lager et program som gjør det for oss.

- **Elevene bruker sin matematiske kunnskap i nye situasjoner**

Elevene har brukt grunnleggende regning og regning med variabler, men i en (for de fleste) ny kontekst. «sum = terning1 + terning2» er addisjon med variabler.

3 Resonnering og argumentasjon

- **Elevene utformer egne resonnement**
- **Elevene forstår andres resonnement**
- **Elevene begrunner fremgangsmåter**

Hele algoritmen er en lang matematisk tankerekke som elevene har vært med på å utforme. I sitt eget arbeid har de vært nødt til å dykke inn i algoritmen og gjøre endringer. For å få det til krever det at de forstår hva vi har gjort sammen, og i tillegg legger til egne instrukser. Dette tolker jeg som å utforme og forstå matematiske resonnement. Mange av elevene begrunnet sine endringer i koden da de oppdaget feil. F.eks. var det et par som hadde et program hvor alle elleve hester flyttet seg fremover på likt. Da gikk de på jakt etter feilen i algoritmen sin og oppdaget at fordi de hadde benyttet den samme setningen på alle hestene «if (terning1 == 1) hestX += 1» ville alle hestene flytte seg ett hakk frem når terning1 viste 1 øye. De begrunnet da at ikke bare måtte de endre på tallet bak «==», de måtte også endre fra «terning1» til «sum» da det var dette de ønsket.

4 Representasjon og kommunikasjon

- **Elevene representerer matematiske begrep, sammenhenger og problemer**
- **Elevene bruker matematisk språk**
- **Elevene veksler mellom ulike representasjoner**

I dette opplegget har elevene representert en fysisk terning som en virtuell terning i et programmeringsspråk ved hjelp av variabler. Elevene har brukt et matematisk språk i opprettelsen, tilordningen og bruken av variablene. At det er ulike sannsynligheter for summen av øynene ved kast av to terninger ble tydelig representert ved hjelp av spydformasjonen til hestene i hesteløpet når programmet var ferdig utviklet. Den skriftlige algoritmen og den visuelle framstillingen er to ulike representasjoner elevene har vekslet mellom i sitt arbeid.

5 Abstraksjon og generalisering

- **Elevene formaliserer tanker, strategier og/eller matematisk språk**
- **Elevene oppdager sammenhenger og strukturer som de formaliserer ved hjelp av algebra og/eller andre hensiktsmessige representasjoner**

Elevene har ved å arbeide med denne oppgaven hatt mulighet til å oppdage at det ikke er lik sannsynlighet for alle summene ved kast av to terninger. Denne sammenhengen kommer tydelig fram i visualiseringen av hesteløpet. Elevene har da oppdaget sammenhengen, formalisert det i form av algebraiske koder som gir en visuell

representasjon. Et terningkast er også blitt abstrahert; 6 muligheter med lik sannsynlighet.

6 Matematiske kunnskapsområder

- **Elevene arbeider med fagstoff fra de matematiske kunnskapsområdene**

«Kunnskap om statistikk og sannsyn gir elevane eit godt grunnlag når dei skal gjere val i sitt eige liv, i samfunnet og i arbeidslivet.» (Utdanningsdirektoratet, 2020b)

Elevene har arbeidet både med statistikk og sannsynlighet. Hesteløpet er en visualisering av en frekvenstabell; hvor ofte inntreffer en hendelse. Sannsynligheten for et gitt antall øyne på en terning anslås til å være 1 av 6. Gjennom dette opplegget får elevene innsyn i både teoretiske sannsynlighetsmodeller; hvor sannsynligheten kan beregnes som antall gunstige utfall dividert på antall mulige utfall, og i frekvensbaserte sannsynlighetsmodeller; hvor den relative frekvensen kan brukes til å anslå en sannsynlighet. Etter å ha gjennomført dette undervisningsopplegget er det også naturlig å snakke om «den store talls lov».

4.4 Vurdering av undervisningsopplegg

Her vil jeg ta utgangspunkt i de fire fordelene jeg ønsket å planlegge undervisningen min med og vurdere i hvor stor grad jeg har lykkes med disse målene.

4.4.1 Scratch – Mangekanter og vinkler

Da jeg lagde dette undervisningsopplegget hadde jeg fritt spillerom. Jeg kunne velge et hvilket som helst tema og gjennomføre det akkurat slik jeg selv ønsket. Jeg valgte Scratch som programmeringsspråk både fordi det var kjent for meg, men også fordi jeg mener at det er intuitivt og brukervennlig selv på lavere alderstrinn. Jeg valgte geometri som tema fordi det var her jeg så for meg at programmering hadde mest å tilby matematikkundervisningen i skolen.

1) LIST

De to utvalgte elevene til dette opplegget, Kari og Trine, var over gjennomsnittet interessert i matematikk vil jeg si, men hadde lite erfaring med programmering og Scratch. De klarte å komme i gang med oppgavene uten noe hjelp bortsett fra den felles introduksjonen med hele klassen. Det var lav inngangsterskel til oppgavene. Blokkene var intuitive å bruke og ryddig delt inn i kategorier med ulike farger.

Underveis i opplegget oppdaget elevene mønster og sammenhenger som de kunne generalisere og abstrahere med algoritmene sine. At antall gjentakelser multiplisert med antall grader alltid ble 360° (Tabell 6: Systematisere funn) har elevene utnyttet når de har programmert et program som kan tegne en hvilken som helst mangekant. Det var stor takhøyde i opplegget.

2) Konseptuell overraskelse

Elevene nådde målene som jeg hadde satt med undervisningsopplegget. De fant ut at i en regulær n-kant vil nabovinkelen til den innvendige vinkelen alltid være 360° dividert på n. Dette var noe elevene ikke hadde tenkt på før. Helt på slutten av intervjuet ba jeg dem oppsummere hva de hadde lært:

Kari [29:41] Men, vi har lært hvordan vi skal lage mangekanter, egentlig, i hvert fall på det. [30:28] For når man skal ha en 8-kant, så er, ehm, for hver gang liksom så vrir han seg fra den veien og hen der, eh, 360 ganger 8. Og da må man gjøre det 8 ganger …

Trine [30:42] 360 ganger 8?

Kari [30:45] 360 delt på mente jeg. Så vri, så henover, så vri og så henover og så vri og så henover … Og sånn fungerer det med alle mangekanter, egentlig, det har jeg aldri tenkt på.

Underveis i opplegget har elevene gjort flere overraskende oppdagelser, og dermed mener jeg at også dette målet med undervisningen var vellykket.

3) Vide vegger

Elevene delte sin begeistring med hverandre og med meg, men hva de gjorde når de kom hjem vet jeg ikke. Det jeg kan si er at det var god stemning på grupperommet og elevene gav uttrykk for at de hadde lært noe da de forlot timen. De diskuterte da hvordan de kunne lage et program for å tegne stjerner i Scratch. Jeg antar at timen blir snakket om hjemme. Det jeg ikke tenkte over på forhånd var at elevene arbeidet på min maskin som jeg hadde gjort klar på forhånd. De hadde ikke opprettet egen Scratch-bruker og dermed hadde de ikke tilgang til koden sin på egen datamaskin uten å skrive alt om igjen. Det tror jeg de fint hadde klart.

4) Medvirkning

Noe av det første elevene gjorde var å endre farge på pennen til rosa, og senere til lilla. Elevene var aktørene og kunne gjøre endringer i algoritmen som de ville. Da de kom til oppgave e) og kunne tegne «andre former» hadde de lyst til å forsøke seg på stjerner, men med de neste oppgavene og målene mine i tankene, dyttet jeg dem i retningen jeg ønsket.

Trine [13:26] (Hvisker) Vi må lage syvkant.

Kari [13:27] Denne syvkanten altså!

Trine [13:30] Må vi lage syvkanten?

TC [13:31] Dere må ikke.

Jeg sa at de ikke måtte lage syvkanten, men de opplevde at det var det jeg ønsket. Bortsett fra denne «dytten» og tabellen jeg lagde for å systematisere funnene deres (Tabell 6: Systematisere funn) var det lite jeg blandet meg inn i deres frie problemløsing. Noen ganger var jeg behjelpelig med noe teknisk, men ellers styrte elevene seg selv. Elevene var produsentene i denne timen og jeg føler derfor også at jeg lykkes med dette målet.

Oppsummert vil jeg si at opplegget fungerte helt etter planen og jeg er veldig fornøyd.

4.4.2 Scratch – Lineære funksjoner

Dette undervisningsopplegget ble til under tidspress. Da jeg nærmet meg fristen jeg hadde satt for datainnsamlingen hadde jeg fortsatt ikke klart for meg hvordan undervisningsopplegget skulle bli. Jeg visste jeg ønsket å ha fokus på variabler fordi dette også er et område hvor jeg mener programmering har mye å tilby matematikkundervisningen i skolen. Jeg fikk beskjed fra klassens lærer at jeg gjerne kunne starte opp emnet lineære funksjoner, men nøyaktig hvordan jeg skulle gjøre det slet jeg med å finne ut av. Jeg testet ut flere idéer, men følte at det meste kunne bli for

komplisert og dermed heve inngangsterskelen for elevene. Innholdet måtte i tillegg kunne gjennomføres i løpet av 60 minutter. «Funksjonsmaskiner» var en tilnærming jeg hadde brukt i min vanlige undervisning, og valgte derfor å digitalisere og tilpasse et slikt opplegg for å inkludere programmering i Scratch.

1) LIST

De utvalgte elevene til dette opplegget var allerede godt kjent med Scratch, så de opplevde at det var lav inngangsterskel til oppgavene. For å sikre lav inngangsterskel for alle elever viste og gav jeg elevene eksempler. Oppgave a) og b) var rutinemessige oppgaver: «Se på eksempelet og kopier». Tanken min var at elevene da ville lære seg programmet, men jeg ser i ettertid at det ble for lærerstyrt. Frykten min for å heve inngangsterskelen har ført til en senkning av takhøyden og begrenset med spillerom. Oppgavene inviterte ikke elevene til fri problemløsing og er ikke egnet for å skape engasjement. Skulle jeg gjennomført opplegget en gang til ville jeg i det minste endret på ordlyden i de første oppgavene. Gitt elevene flere valgmuligheter og tatt vekk noen av føringene. Poenget med oppgavene var jo ikke at elevene skulle regne ut alderen til brukeren om 10 år, eller kunne sette sammen et fornavn med et etternavn. Poenget var å bli kjent med variabler i programmering og dette kunne jeg løst på andre mer engasjerende måter.

Et annet problem med dette opplegget var den store takhøyden, som jeg også var klar over på forhånd. Oppgavene på arket var ikke særlig utfordrende og det var ikke i selve oppgavene at mulighetene for å utforske lå. Nettopp fordi elevene var så godt kjent med Scratch fra før, ble de ferdige med å lage egne funksjonsmaskiner rimelig fort og jeg fikk derfor tid til å øke takhøyden i opplegget gjennom oppfølgingsspørsmålene elevene fikk.

2) Konseptuell overraskelse

Elevene oppdaget at alle tallparene fra en funksjonsmaskin dannet punkter i et koordinatsystem som lå på en rett linje. De fant ut at linjen og funksjonsmaskinen representerte den samme sammenhengen og utnyttet dette til å veksle mellom de ulike representasjonene. Elevene oppdaget overraskelsen jeg planla for, og kunne arbeide videre med dette i de neste timene.

3) Vide vegger

Elevene har gjennom arbeidet sitt denne timen delt oppdagelsene og opplevelsene sine med hverandre. I overgang fra en oppgave til en annen, viste de hvor lett det var å ta vare på koden sin til senere bruk.

Erik [09:44] Skal vi se, da kan vi egentlig… Hvis vi bare, sånn, så kan vi spare koden (løsner koden fra «når grønt flagg klikkes»)

Erik [10:08] Det er ikke vits i å kaste bort god kode.

De sammenkoblede blokkene de ikke lenger bruker sparer de på og kan se på det selv senere eller dele med andre. Mulighetene for deling er der og i hel klasse ville elevene delt sine funksjonsmaskiner med hverandre.

4) Medvirkning

Ettersom jeg ikke helt nådde opp med LIST-målet, var det begrensede muligheter for å utvide oppgavene og følge dem i selvbestemte retninger. Elevene fikk riktignok være

produsenter av digitalt innhold; de måtte opprette, navngi, tilordne og bruke variabler, men oppgavesettingen mener jeg i for stor grad var lærerstyrt.

Oppsummert vil jeg si at opplegget fungerte etter planen, men at planen kunne vært mye bedre.

4.4.3 p5.js – Sannsynlighet

I arbeidet med å planlegge dette undervisningsopplegget brukte jeg mye tid på å lete etter den riktige idéen. Jeg var misfornøyd med opplegg 2 og ville ikke gjøre de samme feilene igjen. Jeg ville arbeide med et tekstbasert programmeringsspråk og hadde nylig blitt kjent med p5.js. De største utfordringene jeg opplevde var knyttet til målet om LIST. Hva kan elevene få til i løpet av en time med begrensede forkunnskaper i programmeringsspråket? Jeg bestemte meg for at jeg ville simulere noe, og endte på idéen med hesteløpet.

1) LIST

Jeg var på forhånd spent på hvordan elevene ville reagere på introduksjonen. Jeg visste jeg måtte skrive og forklare raskt for at elevene skulle ha nok tid til å løse oppgavene sine. I ettertid ser jeg at det fungerte bedre enn forventet. Jeg antar at den tydelige konteksten; «hesteløp» med terningkast, første runde hvor vi gjennomførte hesteløpet og gjennomgangen av hvordan jeg lagde programmet sørget for lav inngangsterskel for elevene. Alle forstod hva de skulle gjøre og hadde fått nok kunnskaper til å sette i gang.

Alle elevene støtte på utfordringer og flere oppdaget mønster underveis som hjalp dem med å forklare resultatene som oppstod i visningsvinduet da programmet ble kjørt. Nøyaktig hvordan programmet skulle se ut var ikke bestemt; ved å basere seg på koden min hadde de en mal, men de kunne også gjøre egne endringer og følge andre spor. F.eks. endre utseende på hestene og hvilken vei de løp. Jeg tenker derfor at det også var stor takhøyde i opplegget, men elevene må være mer kjent med skriftlig programmering for å kunne utnytte potensialet som ligger der. Etter å ha programmert det andre hesteløpet er det gode muligheter for å bruke resultatet til å diskutere den store talls lov, eller simulere andre situasjoner.

2) Konseptuell overraskelse

«Spyd-formasjonen» var for de fleste elevene uventet og overaskende. Elevene som hadde arbeidet mest med programmeringen av programmet var også de som kunne forklare hvorfor hest7 var så godt trent. Flere gunstige utfall betydde større sannsynlighet for å flytte.

3) Vide vegger

Elevparene snakket mindre med hverandre enn det jeg hadde sett for meg. Jeg tenkte at når ett par hadde funnet en løsning, ville det fort spre seg til de andre parene, men det var i grunn kun det ferdige programmet som parene viste fram til andre. Dermed delte de sin oppdagelse og fikk bekreftet at andre også fant det samme da de ble ferdige. Noen elever som ikke var ferdige kunne likevel forklare spyd-formasjonen og ble inkludert i klassens matematiske oppdagelse. Enkelte elever logget inn og lagret koden sin, men jeg tror de fleste bare lukket det vekk når timen var slutt.

4) Medvirkning

Problemet elevene arbeidet med kunne angripes på flere måter og elevene valgte selv hvor de ville starte. De fleste startet med å opprette den andre terningen, men noen valgte å opprette hestene først. Den ene eleven som gjorde at hestene snudde vei og endret deres utseende til fugle-emojier viser at elevene har mange valgmuligheter og kan styre utformingen av programmet selv. De fleste tenkte nok ikke over mulighetene på grunn av begrensede erfaringer med programmering, men med mer arbeid med slike oppgaver vil mulighetene for medvirkning antakelig bli mer benyttet.

Oppsummert vil jeg si at jeg er fornøyd med opplegget, det fungerte etter planen og elevene overrasket meg med hvor mottakelige de var for tekstbasert programmering. Med flere forkunnskaper vil også takhøyden og medvirkningen økes.

4.5 Holdninger

Det var ikke et uttalt mål på forhånd å se nærmere på elevenes holdninger i arbeidet med programmering. I trådmodellen for matematisk kompetanse er det en egen tråd som heter metakognisjon og selvregulering (senere omtalt som engasjement), dette er som nevnt i teoridelen ikke en del av kjerneelementene, men beskrevet i overordnet del som gjelder alle fag. Jeg hadde i utgangspunktet ikke tenkt å ta det med i dette arbeidet, men i ettertid av utprøvingen av undervisningsoppleggene dukket det allikevel opp som et observerbart fenomen.

4.5.1 Å gjøre feil

Som ungdomsskolelærer opplever jeg at en av de største hindringene for læring jeg møter i arbeid med elevene er at de frykter det å gjøre feil. Elevene lærer mer av å gjøre feil, enn av å lykkes (Boaler, 2016). Elevene må fortsette å arbeide, fortsette å prøve, selv om oppgavene er vanskelige. Hvis elevene er redde for å gjøre feil, hindrer de sin egen læringsprosess.

I programmering er feil en helt naturlig del av prosessen. I alle tre opplegg har elevene kjørt programmene sine for å oppdage og rette på feil som en del av problemløsingsprosessen. Kari og Trines strategi når de stod fast var:

Kari [15:11] Okay, jeg vet hva vi må gjøre … Prøve oss fram. Prøve og feile mye!

Feilsøking er en del av det å være en algoritmisk tenker.

4.5.2 Motivasjon, glede, engasjement

Barneskolelærere har fortalt meg at elevene på første og andre trinn ofte trekker fram matematikk som et av de gøyeste fagene på skolen. Når elevene starter på ungdomsskolen trekker de ofte fram matematikk som et av de kjedeligste fagene på skolen. Programmering kan være med på å opprettholde og øke gleden i faget.

Kari og Trine viste mange tegn på glede og engasjement under intervjuet. De klappet stadig når de fikk noe til, de jublet og strakk hendene høyt i været, de skrøt av seg selv og viste meg stolt hva de hadde fått til. Det er ikke ofte jeg observerer at elevene klapper etter å ha fått til en mer tradisjonell matematikkoppgave.

Trine var ikke så begeistret første gang de tegnet en trekant.

Kari [02:52] Ja! Vi klarte det. Vi er så flinke. Ja!

Trine [02:56] Vi klarte å lage en trekant (ikke like begeistret)

Da de brukte det ferdige programmet sitt til å be brukeren om antall hjørner, og så tegne mangekanten, var tonen en annen:

Trine [26:17] Den ser ut som en trekant (fnis) … Wow, vi klarte å lage en trekant. Også står det der hvor mange grader (peker på skjermen). Wow!

Arbeidet hadde tatt tid, det hadde vært utfordrende, og derfor var det også en annen mestringsfølelse involvert. I programmering er man nødt til å *holde ut,* noe jeg ser på som en forutsetning for å kunne virkelig glede seg over resultatet.

Erik og David viste ikke det samme engasjementet med kroppsspråket sitt, men Erik sa ved et par anledninger, helt uoppfordret og uten kontekst at han likte arbeidsmåten:

Erik [18:30] Det hadde vært mye gøyere å holde på med dette her i matten.

Jeg spurte Erik og David på slutten av intervjuet hva de tenkte om koblingen mellom programmering og matematikk:

Erik [43:21] Det er gøyere med programmering og matte, enn bare matte.

David [43:27] Ja.

Erik [43:27] For i programmering da kan vi jo ha sånn der, lage sånn der maskin og så kan du prøve å finne ut av om det er noen sammenheng, alle de forskjellige svarene, og om du klarer å lage maskin som bare flytter litt på forskjellige svarene som blir gitt. … I stedet for å bare sitte og holde på med masse teori, så kan vi jo sitte og, jobbe med sånn der litt mer praktiske oppgaver og du får for eksempel den her oppgaven hvor du skal lage en maskin, så skal du finne ut hva er sammenhengen mellom det her og om du kan regne det ut på en eller annen måte uten å bruke maskinene, om du kunne se et mønster.

TC [44:10] David, noen tanker?

David [44:11] Eh, ja, at, vi bruker veldig mye sånn teknologi i hverdagen som har veldig mye med programmering å gjøre, så hvis man lærer litt av det på skolen, så er det, kan det være veldig nyttig. Fordi flere jobber i fremtiden sier jo at veldig mye om programmering.

TC [44:33] Hva med matematikk faget nå da? Er det ikke nyttig?

David [44:38] Jo, men det blir mer og mer programmering liksom, så, det er veldig få folk som kan sånn programmeringsspråk.

Erik beskriver måten de har arbeidet på som grunnen til at programmering i matematikktimen har vært gøy. Å finne ut av sammenhenger og se etter mønster er viktig både i programmering og «vanlig» matematikk. David flytter fokuset fremover i tid i sitt svar og peker på fremtidens yrker og krav til kompetanse. Programmering kan for noen gjøre matematikkundervisningen mer relevant og dermed motiverende.

4.5.3 Selvtillit

Det etterspørres mer digital kompetanse i befolkningen, men dagens samfunnsborgere er i stor grad kun konsumenter av teknologien og ikke produsenter. Jeanette Wing (2006) argumenterer for at algoritmisk tenking bør være en grunnleggende ferdighet på lik nivå som lesing, skriving og regning. Erik og David hadde brukt digitale verktøy mye før og var selvsikre i arbeidet sitt. Kari og Trine hadde ikke de samme erfaringene, noe spesielt Kari gav uttrykk for:

Kari [08:10] Derfor stoler man ikke på PC!

Trine [08:11] (ler)

Kari [08:13] Har alltid sagt det.

Trine [08:14] Du gjør ikke.

Kari [08:15] Jeg stoler ikke på PCer!

I oppsummeringen av timen trakk Kari dette fram igjen:

Kari [29:41] Jaa. For jeg … Meg og pcer er ikke gode venner. Hver gang jeg er inne på et sånt programmeringsprogram ender jeg opp med å stoppe det.

Elevene har fått flere erfaringer med digitale verktøy og hva de kan brukes til og fått økt selvtillit gjennom gode opplevelser.

I dette kapitlet har jeg presentert mine resultater og analyser. Jeg har forklart hvordan jeg har utviklet og i ettertid vurdert undervisningsoppleggene, og jeg har vist at elevene både har arbeidet med algoritmisk tenking og kjerneelementene gjennom disse oppleggene. I neste kapittel diskuterer jeg funnene mine.

5 Diskusjon

I dette kapitlet vil jeg diskutere resultatene og funnene fra analysekapitlet. Diskusjonen deler jeg inn i tre deler: I første del diskuterer jeg undervisningsoppleggene; utformingen og gjennomføringen av disse. I andre og tredje del fokuserer jeg på elevarbeidet og skriver først om programmering i klasserommet, og deretter om dybdelæring i matematikk.

5.1 Undervisningsoppleggene

En stor del av arbeidet med denne masteroppgaven har vært å utvikle undervisningsoppleggene. Jeg mener jeg har lykkes i varierende grad. Det første opplegget er jeg mest fornøyd med. Her har elevene fått utforske mer enn i de andre to oppleggene. Jeg hadde et mål med oppgave 2) og 3), men disse oppgavene ventet jeg med. Elevene var ikke nødt til å arbeide med disse. I opplegg 2 og 3 var alle oppgavene styrt mot ett sluttresultat. I ettertid ser jeg at dette var valg jeg burde ha unngått. Tanken var vel at siden det hadde gått så bra med opplegg 1 ønsket jeg å sørge for at alle elevene i alle opplegg skulle oppdage konseptuelle overraskelser. De første to oppgavene av opplegg 2 viser dette tydelig. Poenget med oppgavene var at de skulle bruke teknikker de trengte for å etterpå forstå og bruke funksjonsmaskinen som jeg hadde lagd. Men hvor ble det av utforskingen til elevene?

Begrensingen på en undervisningstime var årsaken til at jeg forhastet meg med opplegg 2 og 3. I opplegg 1 var jeg på forhånd forberedt på at elevene kanskje ikke ville komme lenger enn til: «e) hvilke andre former klarer du å tegne?» Det var først etter at jeg observerte elevenes arbeid at jeg visste at oppgave 2 og 3 kunne løses. Lignende valg burde jeg ha lagt til rette for i de andre to oppleggene; mer tid til elevenes egen utforsking. Ikke alle elevene må oppdage det samme, de kan dele sine oppdagelser med resten både underveis og i etterkant. Den konseptuelle overraskelsen jeg planlegger for bør på forhånd være gjennomtenkt, men jeg må ikke styre undervisningen så detaljert mot dette målet som jeg gjorde spesielt i opplegg 2 om lineære funksjoner.

I første opplegg har programmering bidratt med et nytt syn på mangekanter. Elevene lærte noe nytt på grunn av arbeidsmåten. I de andre oppleggene vil jeg ikke si at det er programmeringen i seg selv som har gitt elevene ny kunnskap, men programmeringen fungerer allikevel som en annen arbeidsform. Spesielt i opplegg 2 er det utspørringen med elevene som gir dem noe å undersøke, ikke oppgavene i seg selv. Akkurat disse elevene, Erik og David, syntes programmering var gøy og derfor likte de undervisningen. Det er nok ikke alle som ville vært like begeistret.

Opplegg 3 ble gjennomført uten at jeg gjennomførte oppgavebaserte intervju. Dette gjør at datainnsamlingen er minimal, men samtidig fikk jeg testet opplegget under mer normale forhold. Det er ikke vanlig at læreren kan sitte hele timen med kun ett elevpar. Noen av valgene jeg foretok denne undervisningstimen plaget meg underveis. Først og fremst valgte jeg å tillate at noen elever ikke kom ordentlig i gang med oppgavene. Hadde dette vært en vanlig undervisningstime ville jeg brukt mer tid sammen med dem, selv om de ikke selv ønsket det. Men fordi jeg var opptatt av å gjøre observasjoner til denne oppgaven, tillot jeg det denne gangen. Noe annet som plagde meg var at jeg ikke gav elevene nok tid til å tenke på egenhånd. Jeg ønsket at elevene skulle komme i mål akkurat denne timen, slik at jeg kunne skrive om det her. Jeg hadde flere par å

observere og var derfor kanskje for kjapp med å gi elevene hint slik at jeg kunne gå videre til neste par. Under de oppgavebaserte intervjuene var jeg mer tålmodig, for her hadde jeg kun ett par å forholde meg til. Hadde jeg ikke skullet skrive om hva elevene gjorde denne timen, er jeg rimelig sikker på at jeg ville vært mer tålmodig også i klasserommet.

Med tanke på planlegging og utvikling av undervisningsopplegg føler jeg at jeg har lært mye gjennom arbeidet med denne oppgaven og har fortsatt mye igjen å lære. Å være forsker og lærer samtidig har vært krevende, Forskeren i meg har ønsket å vise til resultater og jeg har selv begått feilene jeg skriver om at ikke bør gjøres. Jeg har tidvis fokusert for mye på resultatene av undervisningsoppleggene og dermed ikke tilrettelagt nok for elevenes egne læringsprosesser.

5.2 Elevarbeid

I denne delen av diskusjonen vil jeg diskutere elevarbeidet presentert i analysekapitlet sett opp mot teorikapitlet og forskningsspørsmålet mitt: *Hvordan kan programmering bli en integrert del av matematikkundervisningen på ungdomstrinnet?*

5.2.1 Programmering i klasserommet

Jeg har til denne oppgaven utviklet tre undervisningsopplegg med programmering som arbeidsmåte til bruk i matematikklasserommet. Med programmering mener jeg både algoritmisk tenking; en prosess, og algoritmer; et produkt. Manglende programmeringskompetanse i den norske befolkning og viktigheten av den i nåværende og fremtidige samfunn er blant grunnene til å innføre mer programmering i skolen (Balanskat & Engelhardt, 2015; Kunnskapsdepartementet, 2016; NOU 2013:2; NOU 2015:8; Sanne et al., 2016). I mine opplegg har elevene analysert, forstått og løst problemer ved å utvikle og verifisere algoritmer. De har arbeidet som programmerere.

I kapittel 4.2 Algoritmisk tenking analyserer jeg elevenes arbeid med utgangspunkt i beskrivelsene til den algoritmiske tenkeren (Tabell 4, s. 24). Her viser jeg hvordan elevene har arbeidet og at dette er i tråd med det som kalles algoritmisk tenking. Elevene har tilnærmet seg problemene på en systematisk måte; brutt problemer ned til mer håndterlige delproblem som løses først, for så å pusle alt sammen til algoritmer som løser de opprinnelige problemene.

Algoritmer sammenlignes ofte med oppskrifter. En kakeoppskrift kan følges, steg for steg, og sannsynligvis resultere i en kake. Smaker den godt er det et godt resultat og oppskriften spares på. Smaker den ikke godt er det et dårlig resultat og neste gang ser en seg om etter en ny oppskrift. Å bruke en algoritme krever ikke all verdens av forkunnskaper. Vi gjør det alle hver dag uten å tenke over det. Men å forstå hvorfor algoritmene fungerer krever mer av oss. Å bruke elevenes mangekantprogram fra opplegg 1 er ikke vanskelig. Trykk kjør, skriv inn hvor mange hjørner du vil at mangekanten skal ha, så ser du resultatet. Men hvordan programmet virker krever mer kunnskap om mangekanter og vinkler som elevene har tilegnet seg i løpet av undervisningstimen. I det siste opplegget hvor elevene skulle simulere et hesteløp med to terninger kunne alle se og bruke programmet. Trykk kjør, så ser en at hest7 stikker av med seieren. Men hvorfor hest7 vant var det elevene som hadde lagd programmet som kunne forklare. Elevene *bruker* ikke bare algoritmene, de *skaper* dem. Samfunnet etterspør flere produsenter av digitalt innhold og der har vi i skolen en jobb å gjøre.

Programmering er en prosess som skaper et produkt. I opplegg 1 kunne jeg ha forklart og vist på en tavle at en kropp som spaserer en runde langs omkretsen til en hvilken som helst mangekant totalt vrir seg 360°. Elevene ville ha tilegnet seg noe av den samme kunnskapen som elevene i opplegg 1 har tilegnet seg. Produktene ville muligens vært de samme, men prosessene ville vært totalt forskjellige. I arbeid med programmering er det viktig å huske på prosessene som settes i gang hos den algoritmiske tenkeren og ikke bare produktene.

Elevene som arbeidet med opplegg 2 var godt kjent med Scratch som programmeringsspråk og støtte ikke på noen tekniske utfordringer de ikke kunne håndtere selv. I arbeid med opplegg 1 og 3 var det flere eksempler på feil som skyldes programmeringsspråket mer enn elevenes intensjoner. Et eksempel fra tekstbasert programmering var da elever skrev kode etter siste krøllparentes. Krøllparentesene indikerer hvilken kode som tilhører hvilken funksjon. Hvis du skriver kode som skal inn i funksjonen «draw», men skriver dette etter krøllparentesen, vil ikke p5.js forstå dette.

```
function draw() {
  background(220);
}
kan ikke skrive kode her
```

Figur 43: feil i p5.js

Et annet eksempel som ble presentert i opplegg 1 var da katten gikk så mange steg at den gikk utenfor visningsvinduet. Elevene trodde de hadde skrevet feil antall grader i koden sin. Dette er eksempler på hvor viktig det er at læreren er kjent med programmene han/hun underviser. Dette kunne forårsaket fryktelig mye frustrasjon om ikke jeg hadde erfaringer med programmeringsspråkene. Betydningen av gode veiledere med kunnskap om programmeringsspråket i bruk er blitt påpekt i tidligere forskning om LOGO og annen programmering i matematikk (Forsström & Kaufmann, 2018).

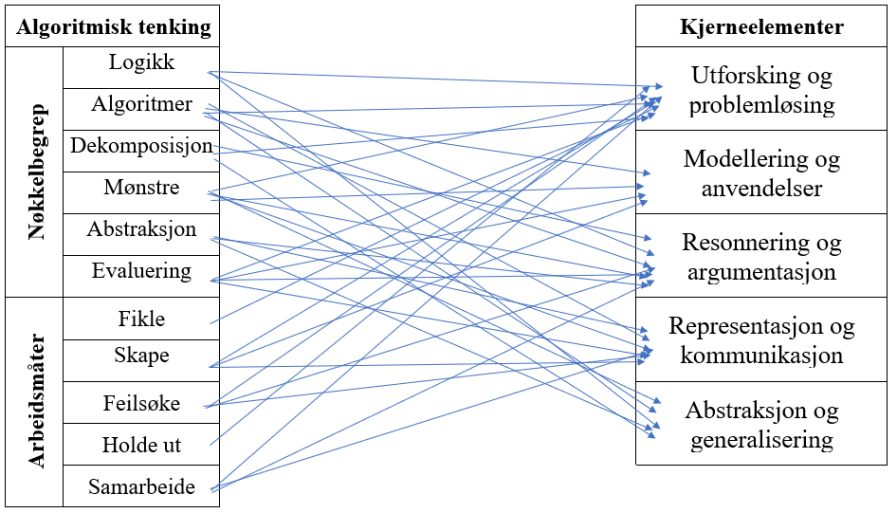
At programmering blir en del av matematikkfaget ser jeg på som en styrke. Jeg blir stadig overrasket over hvor lite digital kompetanse elevene mine har. De bruker teknologiske verktøy i mye større grad enn jeg selv gjorde på deres alder, men har mindre forståelse for hvordan de fungerer. Kanskje brukergrensesnittene har blitt for enkle å forholde seg til; for brukervennlige? Digital kompetanse har i lengre tid vært sterkt prioritert i flere lands utdanningspolitiske dokumenter, men forskning tyder på at de fleste lærere fortsatt i stor grad bruker digitale verktøy hovedsakelig til planlegging og presentasjon av undervisning (Zagami et al., 2018). En går glipp av det enorme potensialet som ligger i teknologien til å hente fram prosessene til den algoritmiske tenkeren. Elevene kan og bør være mer aktive i læringsprosessene. Programmering åpner for at elevene kan bli mer utforskende, skapende, kreative, logiske og samarbeidende. Hvis vi lykkes med implementeringen lover dette godt for fremtiden.

5.2.2 Dybdelæring i matematikk

Jeg oppsummerte kapittel 2.2 Dybdelæring i matematikk ved å forenklet si at hvis en bruker arbeidsmåtene, metodene og tenkemåtene som ligger i de fem første kjerneelementene til å tilegne seg fagstoff som er beskrevet i det sjette kjerneelementet, kan en oppnå dybdelæring i matematikk.

Datamaterialet mitt har jeg analysert i to omganger. Først for å undersøke om det virkelig er programmering elevene arbeider med, deretter for å undersøke om elevene også arbeider med tenkemåtene og arbeidsformene som beskrives i kjerneelementene. I analysen min kommer det tydelig fram at dette er tilfellet. Elevene arbeider både med

programmering *og* kjernen i matematikkfaget samtidig, kompetansene utvikles parallelt.



Tabell 9: Algoritmisk tenking og de fem første kjerneelementene

Min forskning viser, slik det også argumenteres for i teoridelen, at det er stor overlapp mellom algoritmisk tenking og de fem første kjerneelementene i matematikk. Mange av de samme episodene fra analysene mine som viser algoritmisk tenking viser også arbeid med kjerneelementene. Algoritmisk tenking er spesifikt nevnt i beskrivelsen av det første kjerneelementet, men knytter naturlig til seg de andre kjerneelementene også. Elevene skal møte de matematiske kunnskapsområdene gjennom arbeidsmåtene, metodene og tenkemåtene i de fem første kjerneelementene. Programmering kan dermed være en glimrende måte å arbeide med matematikk på.

I opplegg 1 og 3 er det gjennom programmeringsoppgavene at elevene arbeider med kjerneelementene og kan oppnå dybdelæring. I prosessen med å skape programmene, har de måttet sette seg dypt inn i matematikken som ligger bak og beskrive nøyaktig hva det er de ønsker skal skje. I opplegg 2 er det ikke selve programmeringen som legger til rette for dybdelæring, men de muntlige oppgavene elevene fikk under utspørringen. Her kunne en nok oppnådd den samme matematiske tenkingen uten å bruke programmering, men opplegget viser uansett at programmeringskompetansen kan utvikles samtidig som annen kompetanse i matematikk.

Det jeg savner i beskrivelsen av kjerneelementene er det Kilpatrick et al. så fint tydeliggjør i trådmodellen og som Matematikksenteret også presiserer: De ulike komponentene (trådene) er alle viktige og skal støtte hverandre og utvikles parallelt. Diagrammene mine (Diagram 1 s. 78 og Diagram 2 s. 85) viser den samme tankegangen; hvordan det arbeides med de ulike elementene samtidig. Drivkraften er det første kjerneelementet: Utforsking og problemløsing. Gjennom utforskende og problemløsende arbeid flettes de andre kjerneelementene naturlig inn. Jeg har sammen med kollegaer uttrykt bekymring for om noen lærere kanskje tenker å fokusere på ett og ett kjerneelement om gangen: «I dag arbeider vi med problemløsing, neste uke arbeider vi med resonnering». Jeg hørte en skoleleder foreslå å dele skoleåret inn i seks deler, hvor ett kjerneelement har hovedfokus om gangen. Slike misoppfatninger kan lett oppstå om en ikke setter seg inn i hva som menes med dybdelæring i matematikk. Jeg savner derfor formuleringen som ble fjernet fra den første offentliggjøringen av

kjerneelementene: «De fem første kjerneelementene beskriver arbeidsmåter, metoder og tenkemåter i matematikk. Det sjette kjerneelementet beskriver de sentrale kunnskapsområdene i matematikk. Elevene skal møte det sjette kjerneelementet gjennom de fem første kjerneelementene» (Kunnskapsdepartementet, 2018a, s. 15). De sentrale kunnskapsområdene i matematikk skal tilegnes gjennom arbeidsmåtene, metodene og tenkemåtene beskrevet i de første fem kjerneelementene. Det betyr ikke at en må arbeide med alle kjerneelementene i alle timer. Det vil være helt naturlig at elevenes egen utforsking bestemmer hvilke kjerneelementer som får mest oppmerksomhet i undervisningstimen. Nøkkelen for å lykkes tenker jeg ligger i det første kjerneelementet. Det er gjennom gode problemer og utfordringer vi kan legge til rette for at elevene utvikler dybdelæring i matematikk i tråd med kjerneelementene og jeg har vist at programmeringsoppgaver kan egne seg til dette.

I kapittel 4.2 Algoritmisk tenking viser jeg hvordan elevene har arbeidet som programmere; de har brukt arbeidsmåter knyttet til den algoritmiske tenkeren og produsert algoritmer. Mange av de samme utdragene fra elevenes oppgaveløsing som viser algoritmisk tenking brukes også i kapittel 4.3 Kjerneelementene hvor jeg knytter elevenes arbeid opp mot kjerneelementene i matematikk. Det at de samme situasjonene kan brukes til å vise algoritmisk tenking *og* arbeid med kjerneelementene var nettopp det jeg ønsket å undersøke. Hvis elevene arbeider med begge deler samtidig utvikles programmeringskompetanse og kompetanse i matematikk parallelt.

Programmering behøver ikke å være noe som kommer *i tillegg* til annet fagstoff, slik mange fagmiljøer har uttrykt bekymring om (Utdanningsdirektoratet, 2017b, 2017c). Arbeidsmåten kan derimot gi nye innfallsvinkler og føre til større innsikt blant elevene. Gjennom arbeidet med mangekanter og vinkler oppdaget Kari og Trine sammenhenger de ikke kjente til fra før i forbindelse med geometri. Nabovinklene til alle de innvendige vinklene i en mangekant summert blir alltid 360°. Innvendige vinkler og vinkelsummer i mangekanter hadde de lært om i undervisningen tidligere, men erfaringene med Scratch gav en dypere forståelse. Seymour Papert (1980) sa at LOGO-programmering med skilpadden var lettere for elevene å forstå ettersom de kan bruke sin egen kropp som referansepunkt. Det samme gjelder for katten i Scratch.

I alle undervisningsoppleggene ble det naturlig for elevene å benytte seg av variabler. Algebra har vært et problemområde for norske skoleelever. Gjennom programmering kan en få et annet forhold til variabler og nytten av dem. Her brukes variabler bevisst for å oppnå noe en ikke kunne få til før, «[…] the concept empowers the child, and the child experiences what it is like for mathematics to enable whole cultures to do what no one could do before» (Papert, 1980, s. 74). Å simulere tusen terningkast ved hjelp av variabler i p5.js er mer effektivt enn å gjennomføre terningkastene i virkeligheten. Å lage et program som tegner forskjellige mangekanter etter ønske fra brukeren krever at elevene forstår hva som er de essensielle egenskapene til mangekanter og hvilke verdier som må endres på fra mangekant til mangekant. Verdier som varierer krever variabler; variablene tjener en rolle som er meningsfull for elevene. Den tradisjonelle undervisningen i skolen har vært preget av tekniske manipulasjoner av algebraiske uttrykk, noe som kanskje ikke er så viktig i fremtiden da datamaskiner kan utføre disse manipulasjonene for oss (Usiskin, 1988). Variablene som brukes i disse manipulasjonene tjener ofte ingen praktisk rolle. Elevene ser ikke relevansen.

Når elevene ser verdien av det de holder på med, vil det øke motivasjonen og den faglige læringen (Kunnskapsdepartementet, 2016, s. 14). Å arbeide med programmering kan legge til rette for arbeid med kjerneelementene i matematikk. Et av kjerneelementene var abstraksjon og generalisering. Slik variabler brukes i programmering passer det rett inn i dette kjerneelementet. «The essence of computational thinking is abstraction» (Wing, 2008).

I teorikapitlet presenterte jeg kort tidligere forskning på bruken av programmering i klasserommet. Noe av kritikken mot bruken av f.eks. LOGO var at det ikke var observerbar forskjell i de kognitive ferdighetene til elevene som hadde arbeidet med programmering og elevene i kontrollgruppen som ikke hadde arbeidet med programmering (Pea, 1987). I denne oppgaven argumenterer jeg ikke for at programmering som arbeidsmåte er *bedre* enn andre arbeidsmåter. Jeg argumenterer ikke for at programmering skal være kjernen i all matematikkundervisning, men jeg viser at det *kan* læres samtidig som annen kompetanse i matematikk.

I dette kapitlet har jeg diskutert og reflektert over resultatene av analysen min og knyttet det til mitt teoretiske rammeverk. I neste kapittel avsluttes denne masteroppgaven. Jeg presenterer konklusjonen min, ser på pedagogiske implikasjoner og gjør et tilbakeblikk på hele forskningsprosjektet.

6 Avslutning

Avslutningsvis vil jeg oppsummere forskningsarbeidet mitt og presentere en konklusjon. Videre vil jeg vurdere de pedagogiske implikasjonene av arbeidet mitt og til slutt ta et tilbakeblikk på denne oppgaven.

6.1 Oppsummering og konklusjon

Problemstillingen som har vært drivkraften for denne masteroppgaven er:

Hvordan kan programmering bli en integrert del av matematikkundervisningen på ungdomstrinnet?

Bakgrunnen for valg av tema var inkluderingen av programmering i den nye læreplanen i matematikk og den blandede mottakelsen den fikk. De fleste høringsinnspillene til kjerneelementene i matematikk gjaldt pålegget om programmering. Det ble argumentert for at programmering ville føre til større stofftrengsel og mindre tid til dybdelæring (Utdanningsdirektoratet, 2017c).

Målet mitt ble da å undersøke om elevene samtidig som de engasjerer seg i programmeringsaktiviteter også arbeider med kjerneelementene i matematikk. Hvis undervisningen kan tilrettelegges på en slik måte at programmeringskompetanse og kompetanse i matematikk utvikles parallelt vil ikke programmering være noe som kommer *i tillegg*, men være en integrert del av undervisningen.

For å undersøke problemstillingen min utviklet jeg tre undervisningsopplegg i matematikk med programmering som arbeidsform. Oppleggene omhandler tre forskjellige fagemner som alle kan knyttes til de matematiske kunnskapsområdene i det siste kjerneelementet. To av oppleggene ble gjennomført som oppgavebaserte intervju, mens det siste ble gjennomført i egen klasse uten systematisk datainnsamling.

I kapittel 4.2 Algoritmisk tenking legger jeg fram begrunnelser for hvorfor jeg mener elevene har arbeidet med algoritmisk tenking. I kapittel 4.3 Kjerneelementene begrunner jeg at elevene har brukt arbeidsmetoder og tenkemåter knyttet til alle kjerneelementene. Dette vises også visuelt i Diagram 1(s. 78) og Diagram 2 (s. 85). Når en arbeider med programmering settes prosessene til den algoritmiske tenkeren i gang, og disse prosessene overlapper med beskrivelsene av kjerneelementene i matematikk (Tabell 9, s. 98). Kombinasjonen av algoritmisk tenking og arbeid med kjerneelementene har vist eksempler på hvordan programmering kan bli en integrert del av matematikkundervisningen. Det er selvfølgelig fullt mulig å oppnå et lignende resultat; undervisningsopplegg som er innom alle kjerneelementene, uten at programmering er involvert. Poenget mitt er at dette *kan* gjøres gjennom programmering. Programmering *kan* være en god arbeidsmåte for å lære matematikk og programmeringskompetanse *kan* læres parallelt med andre kunnskaper i matematikk.

De nye kjerneelementene i matematikk ble godt mottatt av skole-Norge da de ble presentert. I denne oppgaven har jeg vist at programmering egner seg for å arbeide med disse kjerneelementene. Jeg håper at mitt bidrag kan være med på å dempe kritikken mot programmering i matematikk og gjøre at fokuset heller rettes mot hvordan det kan integreres som en naturlig del av undervisningen. Jeg har utviklet tre undervisningsopplegg som viser eksempler på hvordan dette kan gjøres. De er på ingen måte ferdigsprikrede fasitsvar på hvordan det skal gjøres, men kan forhåpentligvis være

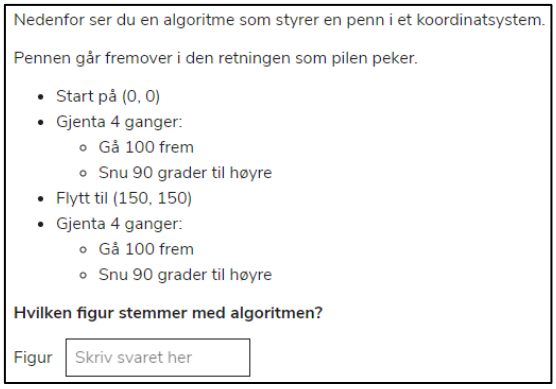
til inspirasjon for andre. Programmering kan by på nye muligheter og innfallsvinkler og kan være med på å berike matematikkfaget. Hvis vi lykkes med innføringen er jeg ikke i tvil om at programmering kan være med på å bidra til dybdelæring i matematikk.

6.2 Implikasjoner

Mine funn viser at programmeringskompetanse og kompetanse i matematikk kan utvikles parallelt og jeg har gitt konkrete eksempler på hvordan dette kan gjøres i klasserommet. Det er viktig å merke seg at jeg i denne oppgaven har definert programmering som todelt; algoritmisk tenking (prosess) og algoritmer (produkt). Alt for ofte hører og leser jeg om henvisning til programmering som et digitalt verktøy. Programmering er mye mer enn det. Det er fullt mulig å arbeide som en programmerer uten å bruke digitale verktøy og det er ikke de ulike programmeringsspråkene som utgjør essensen i programmeringen. Jeg mener det er prosessene til den algoritmiske tenkeren som er kjernen til å lykkes med programmering i skolen. Og der ligger også det store problemet; hvordan fremheve prosessene?

Dessverre (mener jeg) er vi i skolen altfor styrt av sluttvurderinger; vurdering *av* læring; summativ vurdering. Den såkalte wash-back effekten er stor. Det som får mest oppmerksomhet i den daglige undervisningen er oppgavetyper elevene kan møte på eksamen. Jeg har argumentert for at det er prosessene programmering setter i gang som er kjernen til å lykkes med programmering i matematikk, men en eksamen er ikke lagd for å måle elevenes prosesser. Jeg frykter at oppgavetypene som gis på eksamen vil styre programmeringserfaringene elevene vil møte i klasserommet.

Bilde til høyre viser en oppgave fra eksempeloppgaver i matematikk for 10. trinn som viser oppgavetyper elevene kan få på eksamen etter ny læreplan (Utdanningsdirektoratet, 2021). Algoritmen ligner veldig på noe som kunne blitt skapt i undervisningsopplegget mitt om mangekanter og vinkler (4.1.1 Scratch – Mangekanter og vinkler). Men om elevene møter de ferdige algoritmene i undervisningen og ikke får være med på å skape dem, vil mye av læringen, utforskingen og oppdagelsene gå tapt.


Nedenfor ser du en algoritme som styrer en penn i et koordinatsystem.

Pennen går fremover i den retningen som pilen peker.

- Start på (0, 0)
- Gjenta 4 ganger:
  - Gå 100 frem
  - Snu 90 grader til høyre
- Flytt til (150, 150)
- Gjenta 4 ganger:
  - Gå 100 frem
  - Snu 90 grader til høyre

Hvilken figur stemmer med algoritmen?

Figur Skriv svaret her

Bilde 4: Eksempeloppgave programmering

Samtidig må programmeringskompetanse vises på eksamen, ellers vil det ikke bli vektlagt i undervisningen. Så hvordan skal vi lykkes med implementeringen?

Nøkkelen er like åpenbar som den er vanskelig å realisere; læreren. Jeg mener å ha vist at det er mulig å lære programmering og matematikk parallelt. Sentralt i dette arbeidet har vært min egen programmeringskompetanse som jeg i stor grad har opparbeidet meg av egen interesse på fritiden min. I all forskningen jeg har lest om programmering i skolen kommer det fram igjen og igjen at læreren må ha kunnskaper om det han/hun underviser. Det er flere lærere og andre frivillige med programmeringskompetanse som lager og deler opplegg. Det finnes gode nettressurser og kurs. Men hvis ikke lærerne får *tid* til å sette seg inn i det, så vil ikke kompetansen nå ut til alle.

Implikasjonene av mitt arbeid er at lærere må huske å verdsette prosessene som programmering setter i gang, og ikke bare vurdere produktene. For å bli kjent med disse prosessene og hvordan en kan gjøre det i matematikktimene kreves tid. Lærerne i landet må få tid til å videreutdanne seg. Det bør forskes videre på hvordan dette kan realiseres. Jeg etterspør også mer forskning på samme tema som jeg selv har skrevet om; hvordan kan programmering bli en integrert del av matematikkundervisningen?

6.3 Tilbakeblikk

I det jeg skriver dette delkapitlet nærmer jeg meg slutten på arbeidet med denne masteroppgaven. Det har vært en langvarig og krevende prosess som jeg nå skal se tilbake på.

6.3.1 Valg av tema

Det var flere grunner til at jeg valgte å ta en mastergrad i matematikkdidaktikk. En av grunnene var å oppnå større faglig tyngde i et fag jeg er svært engasjert i. Matematikkdidaktikken jeg hørte, leste og observerte at elevene møtte i skolen stemte i svært liten grad overens med forskningen jeg leste på området. Utforsking og problemløsing var ikke toneangivende i undervisningen og jeg slet med å overbevise andre lærere om at dette var veien å gå. Jeg ønsket å lære mer og utvikle meg selv slik at jeg både kunne bli en bedre lærer, men også en bedre fagutvikler.

Etter at jeg hadde bestemt meg for å ta master i matematikkdidaktikk kom fagfornyelsen og introduserte kjerneelementene. Det ble for meg et stort vendepunkt i fagutviklingsnettverkene jeg var en del av. Nå var det ikke til å stikke under stol at kjernen i faget, hensikten til læreplanen, var måten elevene arbeidet med fagstoffet på.

I utgangspunktet hadde jeg lyst til å skrive om elevenes møte med matematikk i skolen. Problemstillingen jeg til slutt valgte ble til etter noen runder med veilederen min. Programmering ble pekt ut som et område jeg kunne skrive om. Jeg underviste programmering som valgfag, visste at det var tydelige likhetstrekk med matematikk og hadde tanker om hvordan programmering kunne berike matematikkfaget. Men jeg hadde ikke prøvd det ut eller tenkt på hvordan jeg faktisk kunne få det til. Det har jeg nå.

Gjennom arbeidet med denne oppgaven har jeg tilegnet meg nyttig kunnskap om utvikling av undervisningsopplegg. Både hva man bør gjøre, og hva man ikke bør gjøre. Jeg føler at erfaringene jeg har fått og refleksjonene jeg har gjort i ettertid har gjort meg til en bedre lærer. Jeg håper at elevene mine vil nyte godt av dette arbeidet jeg har gjennomført.

6.3.2 Kritikk av metode

Noe som kan kritiseres i min forskningsmetode er hvordan jeg har gjennomført utvalget av elever til de oppgavebaserte intervjuene. For det første er det ikke helt tilfeldig hvilke klasser som har vært aktuelle for forskningen min. Jeg er del av et fagutviklingsnettverk i matematikk og har henvendt meg til lærere jeg kjenner fra før. Jeg har vært i arbeid som lærer mens jeg har skrevet denne oppgaven og har måttet velge tidspunkt som passer overens med min egen undervisningsplan. Elevene fikk vite at det var frivillig å delta i forskningen, noe som kan bety at elevene som meldte seg har ekstra interesse for enten matematikk eller programmering eller begge deler. Likevel

mener jeg ikke at dette forringer kvaliteten på mitt arbeid og heller ikke at det har stor innvirkning på hvordan jeg konkluderer.

Det må også nevnes at mine subjektive verdier og holdninger ubevisst kan ha spilt en rolle i hvordan jeg tolker utsagn og kroppsspråk. Det kan være jeg tror jeg ser tegn på noe, men som en annen person hadde tolket ulikt. I analysen min har jeg plukket fram episoder som jeg mener best belyser det jeg forsker på, andre ville kanskje valgt annerledes. Mot slutten av intervjuene har jeg etterspurt elevenes egne tanker og syn på egen læring for å bedre sikre at jeg tolker dem korrekt, men selv da kan mitt eget engasjement påvirke deres svar. Det kan være de vet hva jeg ønsker de skal svare og svarer deretter. En kommer ikke unna noe påvirkning fra forskeren.

6.3.3 Analysen

Det mest tidkrevende arbeidet med denne oppgaven var analysen. Jeg visste tidlig at jeg ønsket å sammenligne den algoritmiske tenkeren med beskrivelsene av kjerneelementene, men ikke nøyaktig hvordan jeg ville gå fram. Jeg har forsøkt meg flere ganger og endte til slutt med todelingen av analysen. Jeg tenker at dette best illustrerer det jeg forsker på.

Selve analysen har også vært krevende fordi beskrivelsene i operasjonaliseringen ikke er klart definerte. Hva er et mønster? Hva vil det si å fikle? Jeg har noen ganger tolket begrepene vidt. Men selv om jeg kanskje i enkelte situasjoner har brukt vide definisjoner av begrepene, mener jeg uansett at hovedpoengene mine er tydelige og kommer godt fram. I ettertid vurderer jeg om jeg kanskje heller burde operasjonalisert trådmodellen for kompetanse i matematikk i stedet for kjerneelementene da kompetansene i trådmodellen er tydeligere definert?

6.3.4 Koronavirus-pandemien

Det tyngste med denne masteroppgaven har vært påvirkningen fra koronavirus-pandemien. Ikke bare ble datainnsamlingen til opplegg 3 stadig forskjøvet og tilslutt avlyst og heller gjennomført i egen klasse, men også tiden min til arbeidet har gått bort. Jeg har studert deltid og har hatt én studiedag i uken til arbeid med fag og oppgave i tilknytning masterstudiet, men denne dagen har i 2020 og 2021 i stor grad blitt spist opp av merarbeid i forhold til læreryrket. Mesteparten av denne oppgaven har blitt skrevet i sommer-, vinter- og påskeferie, så nå er jeg veldig glad for å være ferdig.

7 Referanseliste

Balanskat, A. & Engelhardt, K. (2015). Computing our future. *Computer programming and coding. Priorities, school curricula and initiatives across Europe. European Schoolnet, Brussels*.

BBC. (2021). *Introduction to computational thinking*. Hentet 13.02.2021 fra https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/2

Bell, J. (2005). *Doing your research project: a guide for first-time researchers in education* (4. utg.). Open University Press.

Boaler, J. (2016). *Mathematical mindsets : unleashing students' potential through creative math, inspiring messages, and innovative teaching*. Jossey Bass Publishers.

Bryman, A. (2016). *Social research methods* (5. utg.). Oxford University Press.

Byrd, I. (2021). *To Differentiate: High Ceilings and Low Floors*. byrdseed.com. Hentet 24. feb 2021 fra https://www.byrdseed.com/to-differentiate-lower-floors-and-raise-ceilings/

Candasamy, T. (2019a). *8. trinns elevers begrepsmessige forståelse av mangekanter gjennom programmering i Scratch*. Universitetet i Agder.

Candasamy, T. (2019b). *Scratch og funksjoner - muligheter og begrensninger*. Universitetet i Agder.

Clements, D. H., Battista, M. T. & Sarama, J. (2001). Logo and Geometry. *Journal for Research in Mathematics Education. Monograph*, *10*, i-177. https://doi.org/10.2307/749924

Clements, D. H. & Sarama, J. (1995). Design of a logo environment for elementary geometry. *Journal of Mathematical Behavior*, *14*(4), 381-398. https://doi.org/10.1016/0732-3123(95)90037-3

Dweck, C. (2012). *Mindset: Changing the way you think to fulfil your potential*. Hachette UK.

Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American psychologist*, *34*(10), 906.

Forsström, S. E. & Kaufmann, O. T. (2018). A literature review exploring the use of programming in mathematics education.

Gadanidis, G., Hughes, J., Minniti, L. & White, B. (2017). Computational Thinking, Grade 1 Students and the Binomial Theorem. *Digital Experiences in Mathematics Education*, *3*(2), 77-96. https://doi.org/10.1007/s40751-016-0019-3

Gilje, Ø., Bjerke, Å. & Thuen, F. (2020). *Gode eksempler på praksis - Undervisning i en-til-en-klasserommet*. Forskning, innovasjon og kompetanseutvikling i skolen (FIKS) ved Universitetet i Oslo. https://www.uv.uio.no/forskning/satsinger/fiks/kunnskapsbase/digitalisering-i-skolen/gepp-rapport--undervisning-i-en-til-en-klasseromme/gepp-rapport_15.05.20_fiks.pdf

Gilje, Ø., Landfald, Ø. F. & Ludvigsen, S. (2018). Dybdelæring - historisk bakgrunn og teoretiske tilnærminger. *Bedre skole : tidsskrift for lærere og skoleledere*, *30*(4), 22-27. https://www.utdanningsnytt.no/files/2019/06/27/Bedre%20Skole%204%202018.pdf

Goldin, G. A. (2000). A scientific perspective on structured, task-based interviews in mathematics education research. I A. E. Kelly & R. A. Lesh (Red.), *Handbook of research design in mathematics and science education* (s. 517-545). Lawrence Erlbaum.

Hibert, J. & Lefevre, P. (1986). Conceptual and procedural knowledge in mathematics: An introductory analyis. *Conceptual and procedural knowledge; The case of mathematics*, 1-23.
Hovde, K.-O. & Grønmo, S. (2020). algoritme. I *Store norske leksikon på snl.no*. Hentet 23. februar 2021 fra https://snl.no/algoritme
Karlsen, L. (2014). *Tenk det! : utforsking, forståelse og samarbeid - elever som tenker sjæl i matematikk : ungdomstrinnet*. Cappelen Damm akademisk.
Kilpatrick, J., Swafford, J. & Findell, B. (2001). *Adding it up : helping children learn mathematics* (N. R. Council, Red.). National Academy Press.
Kunnskapsdepartementet. (2016). *Fag - Fordypning - Forståelse: En fornyelse av kunnskapsløftet* (Meld. St. nr. 28 (2015-2016)). Kunnskapsdepartementet.
Kunnskapsdepartementet. (2018a). *Kjerneelementer i fag*. Kunnskapsdepartementet. https://www.regjeringen.no/contentassets/3d659278ae55449f9d8373fff5de4f65/kjerneelementer-i-fag-for-utforming-av-lareplaner-for-fag-i-lk20-og-lk20s-fastsatt-av-kd.pdf
Kunnskapsdepartementet. (2018b). *Retningslinjer for utforming av nasjonale og samiske læreplaner for fag i LK20 og LK20S*. Kunnskapsdepartementet. https://www.regjeringen.no/contentassets/3d659278ae55449f9d8373fff5de4f65/retningslinjer-for-utforming-av-nasjonale-og-samiske-lareplaner-for-fag-i-lk20-og-lk20s-fastsatt-av-kd.pdf
Kunnskapsdepartementet. (2020). *Overordnet del - verdier og prinsipper for grunnopplæringen*. https://www.udir.no/lk20/overordnet-del
Marton, F. & Säljö, R. (1976). On qualitative differences in learning: I - Outcome and process. *British journal of educational psychology.*, *46*(1), 4-11.
Matematikksenteret. (2020). *Meningsfull matematikk for alle - et samspill mellom praksis, forskning og utvikling*. https://www.matematikksenteret.no/om-senteret/visjon-strategi-og-samfunnsoppdrag-2020-%E2%80%93-2025
National Research Council. (2000). *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*. Washington, D.C: National Academies Press. https://doi.org/10.17226/9853
Nosrati, M. & Wæge, K. (2018). Dybdelæring i matematikk.
NOU 2013:2. (2013). *Hindre for digital verdiskaping*. Fornyings-. administrasjons- og kirkedepartementet.
NOU 2014:7. (2014). *Elevenes læring i fremtidens skole: Et kunnskapsgrunnlag*. Kunnskapsdepartementet.
NOU 2015:8. (2015). *Fremtidens skole: Fornyelse av fag og kompetanser*. Kunnskapsdepartementet.
Papert, S. (1972). Teaching Children to be Mathematicians Versus Teaching About Mathematics. *International Journal of Mathematical Education in Science and Technology*, *3*(3), 249-262. https://doi.org/10.1080/0020739700030306
Papert, S. (1980). *Mindstorms : children, computers, and powerful ideas*. Basic Books.
Papert, S. (1993). *The children's machine : rethinking school in the age of the computer*. BasicBooks.
Pea, R. D. (1987). Logo Programming and Problem Solving.
Ringdal, K. (2013). *Enhet og mangfold : samfunnsvitenskapelig forskning og kvantitativ metode* (3. utg. utg.). Fagbokforl.
Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E. & Voll, L. O. (2016). *Teknologi og programmering for alle - En faggjennomgang med forslag til endringer i grunnopplæringen.* Utdanningsdirektoratet. Utdanningsdirektoratet.

https://www.udir.no/globalassets/filer/tall-og-forskning/forskningsrapporter/teknologi-og-programmering-for-alle.pdf
Sawyer, R. K. (2006). *The Cambridge handbook of the learning sciences*. Cambridge University Press.
Schneider, W. & Artelt, C. (2010). Metacognition and mathematics education. *ZDM*, *42*(2), 149-161.
Sevik, K. (2016). *Programmering i skolen*. Senter for IKT i utdanningen. https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf
Skemp, R. R. (1976). Relational understanding and instrumental understanding. *Mathematics teaching*, *77*(1), 20-26.
Sletten, M. (2019). *En designstudie av rike oppgaver med utgangspunkt i autentiske broer i Agder* [Universitetet i Agder ; University of Agder].
Tofteberg, G. (2019, 27. november 2019). *Algoritmisk tenking i matematikk - gammelt nytt i ny kontekst*. Presentasjon på Novemberkonferansen 2019 - Kickstart på kjerneelementene, Trondheim.
Usiskin, Z. (1988). Conceptions of school algebra and uses of variables. *The ideas of algebra, K-12*, *8*, 19.
Utdanningsdirektoratet. (2017a). *Kjerneelementer - fag i grunnskolen og gjennomgående fag i vgo*. https://www.udir.no/laring-og-trivsel/lareplanverket/fagfornyelsen/kjerneelementer/
Utdanningsdirektoratet. (2017b). Kjerneelementer i matematikk, men hvorfor programmering? https://udirbloggen.no/kjerneelementer-i-matematikk-men-hvorfor-programmering/
Utdanningsdirektoratet. (2017c). Oppsummering av matematikk - andre innspillsrunde. https://udirbloggen.no/13194-2/
Utdanningsdirektoratet. (2019a). *Algoritmisk tenkning*. https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/
Høring - læreplaner i matematikk, (2019b). https://hoering.udir.no/Hoering/v2/343
Utdanningsdirektoratet. (2019c, 29.05.2019). *Mål 1 – barn og unge skal få bedre kompetanse i realfag*. https://www.udir.no/tall-og-forskning/publikasjoner/realfagsbarometeret/mal-1/
Utdanningsdirektoratet. (2020a). *Hva er fagfornyelsen?* https://www.udir.no/laring-og-trivsel/lareplanverket/fagfornyelsen/nye-lareplaner-i-skolen/
Utdanningsdirektoratet. (2020b). *Læreplan i matematikk 1.-10. trinn* (MAT01-05). https://www.udir.no/lk20/mat01-05
Utdanningsdirektoratet. (2021, 15.02.2021). *Eksempeloppgaver i matematikk for 10. trinn*. Hentet 07.04.2021 fra https://www.udir.no/eksamen-og-prover/eksamen/eksempeloppgaver/eksempeloppgaver-i-matematikk-grunnskolen/
Wing, J. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. https://doi.org/10.1145/1118178.1118215
Wing, J. (2008). Computational Thinking and Thinking about Computing. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717-3725. https://doi.org/10.1098/rsta.2008.0118
Wing, J. (2011). Research notebook: Computational thinking—What and why. *The Link Magazine*, 20-23.
Yelland, N. (1995). Collaboration and learning with Logo: does gender make a difference?

Zagami, J., Bocconi, S., Starkey, L., Wilson, J. D., Gibson, D., Downie, J., Malyn-Smith, J. & Elliott, S. (2018). Creating future ready information technology policy for national education systems. *Technology, knowledge and learning*, *23*(3), 495-506.

8 Vedlegg

8.1 Informasjonsskriv

Vil du delta i forskningsprosjektet ”Programmering i matematikk”?

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å undersøke hvordan programmering kan styrke matematikkundervisningen på ungdomsskolen. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål
Programmering blir en del av matematikkfaget når den nye læreplanen tas i bruk fra skolestart i 2020. I denne masteroppgaven ønsker jeg å se på hvordan programmering som arbeidsmåte kan berike matematikkundervisningen i klasserommet. I løpet av skoleåret 2019-2020 skal jeg gjennomføre to oppgavebaserte intervju hvor jeg ønsker å analysere læringsutbyttet til elevene som arbeider med programmeringsoppgaver knyttet til bestemte matematiske tema.

Hvem er ansvarlig for forskningsprosjektet?
Universitetet i Agder er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?
Læreren din har tillatt meg å låne klassen din i en matematikktime. I den timen ønsker jeg å intervjue to elever som samarbeider om å løse oppgaver. Kan du tenke deg å være en av de elevene?

Hva innebærer det for deg å delta?
Hvis du velger å delta i prosjektet, innebærer det at du sammen med en medelev løser oppgaver på PC på et grupperom mens jeg observerer og intervjuer underveis. Jeg vil ta notater, lyd- og videoopptak. Foresatte kan få se oppgaver og spørsmål på forhånd ved å ta kontakt.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykke tilbake uten å oppgi noen grunn. Alle opplysninger om deg vil da bli anonymisert. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg.

Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger
Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket.

- Det er kun jeg som vil ha tilgang til å se/høre video-/lydopptaket.
- Navnet ditt vil jeg erstatte med en kode som lagres adskilt fra datamaterialet.
- Datamaterialet lagres på passord beskyttet PC uten internettilgang

Navnet ditt anonymiseres. Masteroppgaven vil kun fortelle at du er elev på en ungdomsskole.

Hva skjer med opplysningene dine når vi avslutter forskningsprosjektet?
Prosjektet skal etter planen avsluttes 01.06.2021. Innen den tid slettes alle opptak.

Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:

- innsyn i hvilke personopplysninger som er registrert om deg,
- å få rettet personopplysninger om deg,
- få slettet personopplysninger om deg,
- få utlevert en kopi av dine personopplysninger (dataportabilitet), og

- å sende klage til personvernombudet eller Datatilsynet om behandlingen av dine personopplysninger.

Hva gir oss rett til å behandle personopplysninger om deg?
Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra Universitetet i Agder har NSD – Norsk senter for forskningsdata AS vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Hvor kan jeg finne ut mer?
Hvis du har spørsmål til studien, eller ønsker å benytte deg av dine rettigheter, ta kontakt med:

- Universitetet i Agder ved
  - Masterstudent Thomas Candasamy, epost: thomac08@student.uia.no, tlf: 400 70 156
  - Veileder Linda Gurvin Opheim, epost: linda.g.opheim@uia.no, tlf: 988 32 585
- Vårt personvernombud: Ina Danielsen, epost: ina.danielsen@uia.no, tlf: 452 54 401
- NSD – Norsk senter for forskningsdata AS, epost: personverntjenester@nsd.no, tlf: 55 58 21 17.

Med vennlig hilsen

Thomas Candasamy

Masterstudent

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet Programmering i matematikk, og har fått anledning til å stille spørsmål. Jeg samtykker til:

- ☐ at mitt barn deltar i et oppgavebasertintervju
- ☐ at jeg deltar i et oppgavebasertintervju (16-åringer kan samtykke selv)

Jeg samtykker til at opplysningene behandles frem til prosjektet er avsluttet, ca. 01.06.2021.

Elevens navn: _____

(Signert av foresatt til prosjektdeltaker, dato)

(Signert av prosjektdeltaker dersom 16 år, dato)

8.2 Transkriberingsnøkkel

, Komma.

. Punktum.

? Spørsmålstegn.

! Utropstegn.

… Pause på mer enn to sekunder.

(…) Beskrivelse av det jeg ser.

[???] Jeg forstår ikke hva som blir sagt.

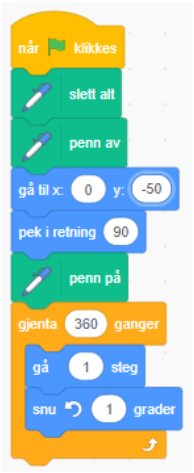
8.3 Oppgaveark: Scratch – Mangekanter og vinkler

Eksempler:

1) Tegne lengde

2) Tegne «stjerne»

3) Tegne sirkel



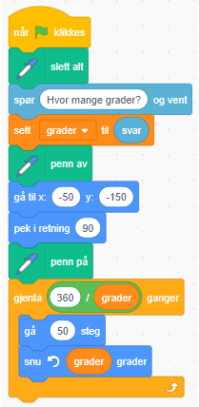
Oppgaver:

1 Løs oppgavene, bestem selv rekkefølge

a) Tegn en trekant
b) Tegn en firkant
c) Tegn en femkant
d) Tegn en sekskant
e) Hvilke andre former klarer du å tegne?

---

2 Hva gjør dette programmet?

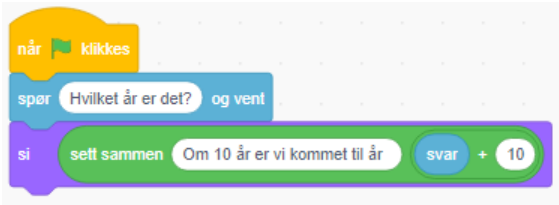


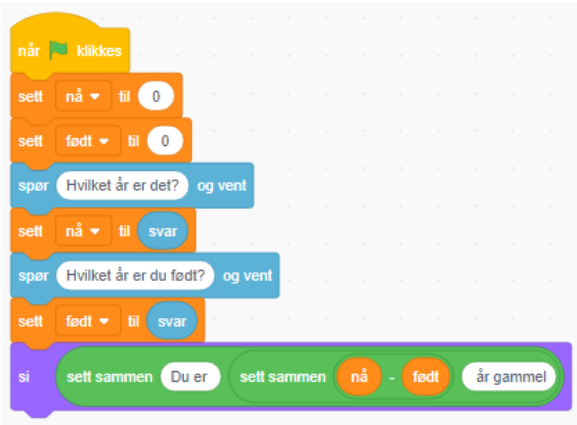
3 Klarer du å lage et program som ber brukeren oppgi antall hjørner, og deretter tegner en mangekant med det gitte antall hjørner?

F.eks:
- Programmet spør: «Hvor mange hjørner?»
- Brukeren svarer: «7»
- *Programmet tegner en 7-kant.*

8.4 Oppgaveark: Scratch – Lineære funksjoner

Eksempler





Oppgaver

a) Lag et program som spør brukeren hvor gammel hun er og som deretter svarer hvor gammel brukeren vil være om 10 år.

b) Lag et program som spør brukeren om fornavnet sitt, deretter om etternavnet sitt og som deretter svarer det fulle navnet til brukeren.

c) Følg linken: https://scratch.mit.edu/projects/341400228/editor/
Fyll inn klosser i de tomme boksene som vil gi ønsket resultat.

d) Gjør din egen endring på «funksjonsmaskinen» slik at maskinen regner ut andre verdier enn originalen. Eksempel:

8.5 Oppgaveark: p5.js – Sannsynlighet

```
var antallOyne = [1, 2, 3, 4, 5, 6];
var terning1;

//"Hestenes" plassering
var hest1 = 0;
var hest2 = 0;
var hest3 = 0;
var hest4 = 0;
var hest5 = 0;
var hest6 = 0;

function setup() {
  createCanvas(640, 480);
}

function draw() {
  background(0);

  //Kast terning
  terning1 = random(antallOyne);

  //Hvilken hest skal flytte?
  if (terning1 == 1)
    hest1 += 1;
  else if (terning1 == 2)
    hest2 += 1;
  else if (terning1 == 3)
    hest3 += 1;
  else if (terning1 == 4)
    hest4 += 1;
  else if (terning1 == 5)
    hest5 += 1;
  else if (terning1 == 6)
    hest6 += 1;

  //Flytt "hestene"
  stroke(255, 255, 255);
  fill(255);
  textSize(50);

  text("1", hest1, height / 11);
  text("2", hest2, 2 * height / 11);
  text("3", hest3, 3 * height / 11);
  text("4", hest4, 4 * height / 11);
  text("5", hest5, 5 * height / 11);
  text("6", hest6, 6 * height / 11);
}
```

a) Hvis vi kaster to terninger og summerer sammen antall øyne på begge terningene, hvor mange ulike summer kan vi få?

b) Hvor mange «hester» må vi ha i hesteløpet vårt?

c) Lag et program som simulerer hesteløpet med to terninger.

d) Kjør programmet!
Hva legger du merke til?
Hvorfor blir det slik?

8.6 Transkripsjon: oppgavebasert intervju om mangekanter og vinkler

		1 Utf. og pbl.	2 Mod. og anv.	3 Res. og arg.	4 Rep. og kom.	5 Abs. og gen.	6 Mat. ksp.
Kari	[00:02] Vi skal tegne ting ikke sant?	1					
TC	[00:04] Og så skal dere lage det i, eh, i Scratch.						
Kari	[00:06] Vi må ha den derre …	1					
Trine	[00:07] Men da …	1					
Kari	[00:08] Det er den.	1					
	(Finner «når grønt flagg klikkes»)						
Trine	[00:08] Ja.	1					
Kari	[00:15] Så må vi ha …	1					
Trine	[00:17] Men da blir det …	1					
Kari	[00:18] Pen på.	1					
Trine	[00:19] Mhmm.	1					
Kari	[00:26] Og så …	1					
Trine	[00:26] Skal vi endre pennens farge (smiler/ler).	1					
Kari	[00:29] Ja!	1					
	[00:33] Ja, vi må ha … Hvilken farge må vi ha?	1					
	(Trine endrer fargen til rosa)						
	[00:40] Ja. Rosa. Da er den rosaaktig.	1					
Trine	[00:42] Sånn.	1					
	(Kloss: sett pennfarge til)						
Kari	[00:46] Ok. Og så … Må den jo gå sånn 100 steg kanskje.	1			4		
Trine	[00:50] Mhm. Kanskje det.	1					
Kari	[00:54] Skal vi skrive det? … Ja!	1					
Trine	[00:56] [???]						
Kari	[00:58] Så må den snu 60 grader, for hvis du skal ha en likesidet trekant så må den jo være 60.	1	2	3	4	5	6
	[01:06] [???] Er det ikke bare sånn gjenta …	1		3	4	5	
Trine	[01:09] Tre ganger.	1		3	4	5	
Kari	[01:10] Mhm.	1		3			
	[01:15] Der er den vett.						
Trine	[01:16] Det er det vi skal.						
Kari	[01:20] Jeg klarer dette.						
	(Trine tar kontroll over datamusen, får gå og snu inn i gjenta klossen. Gå kommer etter snu)						
TC	[01:21] Så er det veldig fint hvis dere, før dere trykker på det grønne flagget, sier hva dere *tror* kommer til å skje.						
Kari	[01:27] Vi tror det kommer til å bli …			3	4		
Trine	[01:28] Vi tror det kommer til å bli en trekant!			3	4		6
Kari	[01:31] Må den ikke gå før den snur? (snu kloss kommer etter gå)	1		3	4		
Trine	[01:34] En likesidet trekant. Eh, jo!	1		3			6
Kari	[01:35] Også tre!			3			
	(Trine skriver at det skal gjentas tre ganger.)						

Kari	[01:37] Vi tror at siden den skal gå [???] steg, så flytter den seg 100 piksler. Og så gjør vi den liten (endrer størrelsen på Felix), sånn.			3	4		
Trine	[01:47] Synes den var litt liten.	1					
Kari	[01:48] Nei, nei… Nå kan vi se at den er der. Så kommer han til å gå 100 steg og så kommer han til å vri seg 60 grader og så kommer han til å gå 100 steg, og så kommer han til å vri seg 60 grader, så kommer han til å gå 100 steg til.	1		3	4		6
Trine	[01:58] Og så blir det en trekant.	1		3			6
Kari	[02:00] Ja, men det kan også være at den ikke vrir seg innover, at han bare blir sånn (viser med fingeren at det ikke blir en trekant).	1		3	4		
Trine	[02:03] Det er sant.	1		3			
Kari	[02:03] Vi kan jo prøve.	1					
	(Trykker på grønt flagg. Tegner ikke trekant.)						
Kari	[02:07] Jaa! (Hendene i været) Det var det som skjedde.	1					
Trine	[02:08] Det ja …						
Kari	[02:09] Så vi må vri han 120 da.	1	2	3	4		6
Trine	[02:14] Vi kan prøve.	1					
Kari	[02:15] Det dobbelte.	1			4		6
Trine	[02:17] Det ble en trekant!	1			4		6
Kari	[02:17] Jaa! (Hendene i været) Det ble det, det var bare det at vi glemte å slette alt.	1					
Trine	[02:24] Vi kan slette før …	1					
Kari	[02:25] Vi kan ta på sånn slett alt ting … Med den	1					
Trine	[02:29] Ja, slett alt. Før alt annet. Ja.	1					
Kari	[02:31] Og så må vi sette han på at han går til den derre … her? (Kloss: gå til)	1					
Kor	[02:38] Null, null.	1			4		6
Kari	[02:42] Og så prøver vi. (Neve i luften, engasjert)	1					
	(Hodet på skrå, ikke helt fornøyd)						
Trine	[02:45] Ja, det ble [???]	1					
Kari	[02:47] Penn av først. Det er riktig, det må vi	1					
Trine	[02:48] Penn …	1					
Kari	[02:50] Eller …	1					
	(Tester program, vellykket) (Kari, hendene høyt i været, kjempe fornøyd.)						
Trine	[02:51] Var det …	1					
Kari	[02:52] Ja! Vi klarte det. Vi er så flinke. Ja!	1					
Trine	[02:56] Vi klarte å lage en trekant (ikke like begeistret)	1			4		
Kari	[02:58] Ja!	1					
Trine	[02:59] En rosa trekant.	1					
Kari	[02:59] Ja … Firkant!?	1					
Trine	[03:02] Åh, ja.	1					
Kari	[03:03] 90 grader.	1			4	5	6
Trine	[03:04] Da blir det 90 grader og så …	1			4	5	6
Kari	[03:05] 90 grader.	1					
Trine	[03:06] Å ja, det er jeg som skal gjøre det … Og så … fire.	1			4	5	
Kari	[03:11] Da tror vi at akkurat samme skjer bare at han går 90 grader istedenfor. Det kan være at vi må endre det til …	1		3			
Trine	[03:17] Vi finner ut av det.	1					
Kari	[03:20] … noe annet	1					
Trine	[03:21] Det ble en firkant. (Tydelig fornøyde) Eh, og så femkant. Da gjør vi … nesten det samme.	1		3	4		6

Kari	[03:27] Er ikke det 115 eller noe? Jeg husker ikke hvor mye det er i en femkant.	1		3		5	6
	(Stillhet, tenker)						
Trine	[03:33] (Ler)						
Kari	[03:33] Jeg husker aldri det.	1					
Trine	[03:34] Ikke jeg heller. Jeg er ikke veldig god på …	1					
Kari	[03:38] Jeg husk … Jeg husker opp til firkanten nettopp var så var den litt sånn ….	1					
Trine	[03:42] (ler) Ganske langt fra firkant.	1					
Kari	[03:44] Vi kan ta … Vi kan jo hvertfall prøve med … Skal vi prøve med 120 sånn bare for gøy for da øker den med 30.	1		3			
Trine	[03:54] Okay. Vi kan prøve … Også kanskje fem ganger. Fem.	1		3			
Trine	[04:02] Det ble en trekant (smiler).	1					
Kari	[04:02] Det var det er sant, det ser vi jo! … Hva var det vi gjorde i stad?	1					
Trine	[04:07] Ehm … Bare …						
Kari	[04:10] Vi må ha det mindre ikke sant? Må vi ikke det?	1					
Trine	[04:12] Hmmm … (ler)						
Kari	[04:16] [???] på en sekskant er det bare til 60.	1		3	4		6
Trine	[04:18] Ja, men på … femkant … for på en firkant er det 360, så da er det … vent, jeg må tenke … (ler) nei …	1		3			6
Kari	[04:31] Fordi, når vi tok 60 i stad så ble det jo liksom starten på (viser med finger) et eller annet. Så hvis vi prøver å ta sånn rundt fem … ti, nei ikke femti, eh … Prøv å skriv 70.	1		3	4		
Trine	[04:43] 70?	1		3			
Kari	[04:44] Ja, det er 10 mer enn 60 … Det er matte.	1		3			
	(Trykker grønt flagg, fungerer ikke helt)						
Trine	[04:49] Eh, jeg tror det skal være.	1					
Kari	[04:51] Det var nærme nok da. (smiler)	1					
Trine	[04:53] Kanskje det er 75?	1		3			
Kari	[04:56] Vent litt … Se … se … se… Det blir sånn, sånn nesten.	1					
Trine	[05:00] Ikke helt.	1					
Kari	[05:01] Det ble [???] godt nok!	1					
TC	[05:04] Er det godt nok?						
Kari	[05:06] Nei, det er ikke det, jeg vet..	1					
Trine	[05:09] 75?	1		3	4		
Kari	[05:10] Prøv 75.	1		3	4		
Trine	[05:11] Oi, eh, du må trykke for meg.	1					
	(Prøver, smiler)						
	[05:16] Det var litt for mye.	1					
Kari	[05:17] Litt mye.	1					
	[05:23] 71	1					
Trine	[05:24] 73?	1					
Kari	[05:26] 72,5	1					
Trine	[05:27] Jaa … Går det an å ta …	1					
Kari	[05:29] Jeg har ikke peiling.	1					
Trine	[05:31] Desimaltall. … Nei det ble ikke noe desimaltall. Tar 73 … Rundet det opp litt.	1					
	(Nesten fornøyd)						
Kari	[05:44] Jeg føler at det nesten går.	1					
Trine	[05:47] Nei, det ble en trekant på slutten … 72 da.	1		3			

Kari	[05:50] Ja, jeg husker bare aldri …	1					
Trine	[05:52] Vet du hva, jeg kan styre selv om du ikke …						
Kari	[05:59] 72 … (prøver et par ganger, engasjement, glede)	1					
Trine	[06:03] Yes, vi klarte det!	1					
Kari	[06:05] Nå har vi lært det!	1					
Trine	[06:08] Sekskant.	1					
Kari	[06:10] Det var 60. Bare at vi må ta det 6 ganger tror jeg. Vi prøver det, for det føltes sånn i stad.	1		3	4	5	6
Trine	[06:15] «Det føltes sånn i stad»?	1		3			
Kari	[06:17] Jeg følte at det var riktig.			3			
Trine	[06:19] Du bare føler det?			3			
Kari	[06:21] «Jeg føler det i meg». Ja, fordi. Ja! Fordi, ehm … En sekskant har dobbelt så mange kanter som en trekant, så det vil gi mening at de er dobbelt så store som en trekant.	1	2	3	4	5	6
Trine	[06:29] Okay, kanskje.			3			
Kari	[06:30] Kanskje. Eller så er det bare sånn rart …	1					
Trine	[06:32] Bare noe du fant på?	1					
Kari	[06:34] Sannsynligvis.	1					
	(Prøver, virket ikke)						
Kari	[06:38] Det er sånn nærme nok da.	1					
Trine	[06:39] Det har syv kanter?	1					
	(Felix traff kanten på vinduet, forstyrrer måling av vinkel. Elevene fant ikke ut av det, begynte å prøve forskjellige vinkler. Jeg ba dem endre start posisjonen til Felix slik at han ikke traff kanten, og gå tilbake til 60 grader)						
Kari	[08:05] Så vi hadde rett da egentlig?	1					
	(Prøver igjen, det virket)						
TC	[08:08] Så det er bare programmet som … tuller litt av og til.						
Kari	[08:10] Derfor stoler man ikke på PC!			3			
Trine	[08:11] (ler)						
Kari	[08:13] Har alltid sagt det.						
Trine	[08:14] Du gjør ikke.						
Kari	[08:15] Jeg stoler ikke på PCer!						
Trine	[08:16] Du er nesten en gammel person.						
Kari	[08:18] Nesten! Jeg er yngre enn deg! … Hvilke andre former klarer vi å tegne? Åh… (ler) Vi skal klare å tegne sirkel … Sannsynligvis da.	1					6
Trine	[08:31] Eh, vi kan …	1					
Kari	[08:32] Vi kan jo prøve å tegne en faktisk stjerne.	1					6
Trine	[08:35] Wow, kan vi det…	1					
Kari	[08:39] Nei, en faktisk sånn (tegner med fingeren).	1			4		
Trine	[08:42] Å, sånn stjerne. …	1			4		
Kari	[08:54] Eh, å ta liksom syvkant og sånn er jo bare å endre på grader greia. Så det blir jo egentlig litt kjedelig.	1		3	4	5	
TC	[09:01] Hvordan kan dere lage syvkanten da?						
Kari	[09:03] Vi må endre på, vi må gjøre sånn at han snur seg …	1		3			
Trine	[09:06] Endre der (peker) også i hvor mange ganger det gjentar.	1		3	4		
Kari	[09:09] Ja, mindre der, og større der.	1		3			
TC	[09:12] Hvor mye mindre må gradene bli da?						
Kari	[09:15] Nei, eh …	1					
Trine	[09:15] (ler) … Hmmm….	1					

Kari	[09:19] Så var det jo det da … En åttekant har eh …	1					
Trine	[09:25] Kan du det?	1					
Kari	[09:27] (ler) En åttekant?	1					
Trine	[09:28] Har det ikke bare dobbelt så m …	1		3	4	5	
Kari	[09:30] Ja, prøv å ta 45 du. Ta 45.	1		3	4	5	
Trine	[09:32] Okay. Ja, da er det 45 og så er det 8 ganger … For da er det … eller er det?	1		3	4	5	6
	(Prøver, Felix treffer kanten, mislykket)						
Trine	[09:39] Nei (smiler), det var ikke det.	1					
TC	[09:40] Det kan være han tuller litt, prøv å flytt figuren igjen. Bytt på x og y verdi.						
Kari	[09:44] Null kanskje? … Null kanskje? Eller så kan vi bare gjøre den … Hvis du endrer den til 75, så blir den ikke like stor.	1		3	4		
	(Prøver, virker, fornøyd, klapper)						
Trine	[10:01] Det er en åttekant.	1					
Kari	[10:02] En åttekant	1					
TC	[10:03] Så hvordan visste du at det skulle være 45 grader?						
Kari	[10:06] Eh fordi, en trekant var 120 og …			3			
Trine	[10:09] Ja! Og så …			3			
Kari	[10:09] Da ble sekskanten 60 … så da … åttekant det er eh dobbelt så mange kanter som en firkant [???] alle er 90	1		3	4	5	
Trine	[10:15] Og da er halvparten så mange … grader.	1		3	4	5	
Kari	[10:18] Mhm. Halvparten av 90. … Så det går ikke an å lage en 16-kant, for da må det være komma (elevene trodde ikke at Scratch godtok desimaltall).	1	2	3	4	5	
TC	[10:24] Men i stad når dere lagde femkanten, så sa dere 72 grader …						
Kari	[10:29] Ja, det er jo bare å ta 36 …	1					
TC	[10:32] … Og så sa dere, nå har vi lært det. …						
Trine	[10:35] Ja, vi kunne det sikkert. Vi bare husker det ikke.	1					
TC	[10:37] Det jeg lurer på … Hva har dere lært? Når det stod 72 grader?						
Kari	[10:41] Nei, vi har lært at eh…						
Trine	[10:42] Vi har lært at det …						
Kari	[10:43] Det er 108 grader i en regulær femkant … ikke sant? … Det blir jo det? For 108 + 72 er 180.	1	2	3	4		6
Trine	[10:52] (Trekker på skuldrene, vet ikke)						
Kari	[10:57] Bare prøv, vi prøver … 7 også 36.	1			4		
Trine	[11:08] Det ble litt …	1					
Kari	[11:08] Det er fordi han gikk litt utenfor sida (ler).			3			
Trine	[11:12] Du kan gjøre det litt mindre (ler) … Du kan ha 50 i stedet for … Så ser vi hva som skjer … Hvis ikke så har vi bare …	1		3	4		
Kari	[11:22] Hvis vi gjør det flere ganger kanskje? (Øke gjentakelser for å lukke figuren)	1		3			
Trine	[11:24] (Begge ler) Da blir det ikke en syvkant.	1		3			
Kari	[11:28] Da blir det en 10-kant.	1					
Trine	[11:33] Tikant da … Tikant!	1					
Kari	[11:37] Vi har lagd en tikant også.	1			4		
Trine	[11:38] Vi klarte å lage en tikant.	1					
Kari	[11:40] Vi er så flinke … Men … fordi syv, det er dobbelt så mye som 3,5 … En trekant var 20 … Nei, 120	1		3	4		
Trine	[11:50] Det er ikke bare 20.	1		3			
Kari	[11:51] Nei, 120, okay? … så da …	1		3			
TC	[11:58] Dette er vel det dere har gjort til nå. (Viser en tabell med mangekant og grader)						

Trine	[12:08] Ehm … Tikant også.	1					
Kari	[12:10] Ja, vi må ikke glemme den!	1					
	(Tegner og skriver tikant inn i tabellen, opptatt av at den skal være regulær)						
TC	[13:15] Så det jeg lurer på er om dere ser et mønster her, som kan hjelpe dere med å lage syvkanten?.						
Trine	[13:26] (Hvisker) Vi må lage syvkant.						
Kari	[13:27] Denne syvkanten altså!						
Trine	[13:30] Må vi lage syvkanten?						
TC	[13:31] Dere må ikke.						
Trine	[13:39] Hmm…						
Kari	[13:40] Og det er 60 …			3			
Trine	[13:41] Ja, men … femkant, tikant, og så kan du se, ja …	1		3			
Kari	[13:45] 40 ganger 3 … Det er ikke noe jeg kan komme på med en gang ganger 4.	1		3			
	[14:04] Når jeg gjør noe, pleier jeg å gjøre det veldig mye vanskeligere for meg selv … enn det det burde ha vært.			3			
	[14:25] Ville en åttekant vært 45?	1			4		
Trine	[14:30] Vi lagde en åttekant også. Du skrev [???]	1					
TC	[14:31] Det er sant, det er sant. Dere lagde den.						
Kari	[14:34] Tegn en åttekant!	1					
	(Trine tegner åttekant inn i tabellen)						
Kari	[15:11] Okay, jeg vet hva vi må gjøre … Prøve oss fram. Prøve og feile mye!	1					
Trine	[15:17] Se, det er det samme med denne og denne fordi det [???]	1		3			
Kari	[15:25] Så vi har ikke noen 3,5 kant her.			3			
Trine	[15:27] Nei, det har vi ikke.			3			
Kari	[15:29] Det er sykt kløne te egentlig … Men, vi kan jo egentlig bare prøve og feile sånn sykt mye … Hvis vi endrer den til 10 ganger, og så bare tar vi alt mellom 60 og 45 … Hvis du finner sånn ca midten. Det vil være 53. [???] 53, så ser vi om det fungerer.	1		3			
TC	[15:57] Hvis dere ser på katten ...						
Trine	[16:01] Ja, skal vi se på katten?						
TC	[16:02] … Når han beveger på seg.						
	(Gjør Felix synlig, tester 53 grader)						
	[16:05] Hvor mange grader …						
Trine	[16:07] (puster ut, oppgitt, mangekanten stemte ikke) 52? Jeg vet ikke, vent … (Retter oppmerksomheten mot TC)	1		3			
TC	[16:11] … går han totalt?						
Kari	[16:13] Litt over 360, ÅÅ! Åååå jaaa … (smiler til Trine) Vi må gjøre sånn at det blir 360.	1		3	4	5	6
Trine	[16:17] For her er det …	1					
Kari	[16:18] 7 delt på … 360 delt på 7.	1	2				
Trine	[16:21] Her er det tre hundre [???] Her er det [???] åtti …						
Kari	[16:26] Nei, men du må jo gange det med 3 … Gange det med 4, gange det med 5, gange det med 6 … Det er 360 delt på 7 … Kan vi skrive på denne?	1		3	4	5	
TC	[16:35] Ja, ja … Skriv i vei.						
Kari	[16:38] Så det blir delt på 7 … Okay, så har vi den gamle måten. Hvor mange ganger får du plass til 7 i tre (ler)?	1		3			6
TC	[16:47] Hvis dere bruker kalkulatoren på maskinen så går det kanskje litt raskere?						
Kari	[16:48] Ja det er sant, er det her det?						
TC	[16:50] Eller så kan dere kalkulere i Scratch forresten, det er litt gøyere.						
Trine	[16:55] Kan vi det?						

	(Viser operatorer i Scratch)						
Kor	[17:24] Ja! Vi klarte det også.	1					
TC	[17:29] Og da kan dere gå videre på oppgave 2 og så oppgave 3 (gir nytt ark).						
Trine	[17:36] (Leser) Hva gjør dette programmet? Ok, la oss finne det ut.	1					
	(Skriver av programmet, finner raskt fram til riktige klosser vha fargekoder og tidligere kjennskap til programmet?)						
Kari	[20:04] Okay, hva tror du kommer til å skje?	1					
Trine	[20:08] (Smiler) Hmmmm.	1					
Kari	[20:09] For vi måtte snakke.						
Trine	[20:12] Må vi snakke? Hva hvis jeg ikke …						
Kari	[20:14] Det her er jo bare en lettere måte å tegne en syvkant på! … Eller er det enda mer komplisert kanskje? (Tester programmet)	1		3			
Trine	[20:20] «Hvor mange grader?»	1					6
Kari	[20:41] Skriv 3,5!	1					
Trine	[20:42] Shhhh…						
Kari	[20:43] Ja, men jeg vil finne ut av hvordan en 3,5 kant ser ut … Nei, ja … Klar?	1					
Trine	[20:52] Ja!	1					
Kari	[20:54] Oi!	1					
Trine	[20:55] Det var litt …	1					
Kari	[20:56] Jeg tror ikke det ble riktig … Vi prøver igjen … Du bestemmer.	1					
Trine	[21:03] Hmm… 42	1					
	(Tegningen gikk utenfor kanten)						
Kari	[21:18] Vi tar sånn 12 steg (endrer i koden), eller 25.	1					
	(Tegningen gikk utenfor kanten)						
Trine	[21:42] Det blir fortsatt samme tingen, det er bare fordi …			3			
Kari	[21:44] Nei, det er fordi vi begynner så langt nede …			3			
Trine	[21:45] Jaa.			3			
Kari	[21:47] Men den herre eksempeltingen er så kløne te. (De snudde motsatt vei)			3			
	(Endrer x og y til 50, tester)						
Trine	[22:02] Nesten.	1					
Kari	[22:02] Ja	1					
Trine	[22:03] Nei, du kan se der at den er litt lengre …	1					
Kari	[22:06] Men det later vi som om ikke eksisterer der… Okay?	1					
TC	[22:19] Prøv å gå videre på oppgave 3.						
	(Leser oppgaven sammen. Kikker på koden de har foran seg.)						
Kari	[22:49] Her er det … ehm … liksom gradene. Han snur 360 delt på de gradene. Sånn at hvis du skriver 50, nei … 36, for det er greit. Så blir det jo.	1	2	3	4	5	6
Kor	[23:08] Ti!	1					
Kari	[23:11] Så da gjentar han det 10 ganger. Og da går han 25 steg, så snur han, hvor mange, jo, 36 grader. Går 25 steg, snur 36 grader … 10 ganger.	1		3	4		
	[23:29] Så hvis vi … vi må først endre den [???] sånn at den heter hjørner. (endrer navn på variabelen fra grader til hjørner)	1		3		5	
	[23:57] F.eks. den med 7-kant. Da ender han jo opp med å måtte … Kanskje vi ikke trenger den (peker på variabelen som angir antall grader)? Nei, jo det gjør vi.	1		3			
	[24:07] Men da må vi ha sånn ekstra greie, må vi ikke det?	1					
Trine	[24:09] Sånn ekstra greie?	1					
Kari	[24:10] Ja, det må vi … Tror jeg … Jeg har ikke peiling.	1		3			

Trine	[24:13] Vi kan prøve … Men hvilken ekstra greie er det du tenker på?	1		3			
Kari	[24:17] Fordi, hvis vi legger inn sånn operator ting. Også tar vi …	1					
Trine	[24:20] Ehm, og da er det …	1					
Kari	[24:24] ... liksom … For hvis vi lager en ekstra sånn variabel ting. Som heter, ehm, grader da. Ikke sant? Så sier den hvor mange grader den skal snu. Og de gradene er 36 delt på hjørner. Sånn at den snur ikke «hjørner» den snur «grader», ikke sant?	1	2	3	4	5	6
Trine	[24:43] Så tar vi …	1					
Kari	[24:44] Så hvis vi … tar …	1					
	(Trenger hjelp til å lage en ny variabel)						
Kari	[25:23] Ok, den kan vi kalle grader.	1				5	
	[25:30] Og grader er … da ehm (finner riktig operator kloss)… Gradene er 36 … 306, 360 mener jeg … delt på svaret … Sånn ikke sant, svar.	1		3	4	5	
Trine	[25:49] Ja.			3			
Kari	[25:52] Og den skal snu, på variabler, så skal den snu … grader. Ikke sant?	1		3			
Trine	[26:00] Tror det. La oss prøve.	1		3			
Kari	[26:02] Eh, bare endre den [???] Så blir det ordentlig.	1					
	[26:08] Jeg vet ikke om dette kommer til å fungere. Skal vi prøve?	1					
Trine	[26:10] Ja.	1					
	(Tester)						
Kari	[26:13] Ja, hvor mange hjørner? Okey, 3? Vi vet hvordan den skal se ut.	1					
Trine	[26:17] Den ser ut som en trekant (fnis) … Wow, vi klarte å lage en trekant. Også står det der hvor mange grader (peker på skjermen). Wow!	1					
Kari	[26:27] Se så … Jadda! … 7	1					
	(Prøver programmet med 7 som input)						
Trine	[26:32] 7 … Og da er det … 51,428571	1					
Kari	[26:41] Det her er som pi det … Veldig mye, ja nå har vi lært det.			3			
	[???] (Endrer antall steg i koden, tester på nytt. Virker ikke slik de forventer …)						
TC	[28:00] Hvis dere kikker litt på den derre gjenta klossen deres.						
Trine	[28:04] Gjenta …						
Kari	[28:05] Ja, den …						
TC	[28:06] Hvor mange ganger skal han gjenta?						
Kari	[28:07] 360 delt på hvor mange hjørner det skal være … Men det trenger vi jo ikke å ha.	1		3			
Trine	[28:12] Nei, vi kan ta vekk den …	1		3			
Kari	[28:16] Gjenta sånn 850. Nei, vent litt …			3			
Trine	[28:20] 850 ganger (ler)			3			
Kari	[28:21] Gjenta hjørner, ikke sant?	1		3	4	5	
Trine	[28:23] Ja. Åh!			3			
Kari	[28:25] Ja, mhm.	1		3			
Trine	[28:26] La oss prøve.	1					
Kari	[28:27] 56 … eller 42 … eller 56.	1					
	(Tegner utenfor vinduet, får en slags spiral)						
Kari	[28:34] Okay, vi hadde egentlig klart det, det er bare at den er for stor.	1		3			
Trine	[28:37] Okay, la oss gjøre den litt mindre. (endrer steg til 12)	1					
	(Vellykket test, Trine klapper)						
Kari	[28:53] 56 kaant.	1					
Trine	[28:55] Og da er det 6,428571	1					

Kari	[28:58] Det gidder ikke jeg prøve å huske.						
Trine	[29:00] Nei, ikke jeg heller, men …						
Kari	[29:02] (Snur seg mot TC) Vi lagde en 56-kant (stolt).						
TC	[29:03] Dere lagde en 56-kant (bekreftende)						
Kari	[29:05] Mhm, det må du skrive ned … 42-kant? … Da har vi klart det da, egentlig.	1					
TC	[29:33] 30 sekunder igjen av timen. Hvis dere kan prøve å oppsummere, føler dere at dere har lært noe av dette?						
Kari	[29:41] Jaa. For jeg … Meg og pcer er ikke gode venner. Hver gang jeg er inne på et sånt programmeringsprogram ender jeg opp med å stoppe det. Men, vi har lært hvordan vi skal lage mangekanter, egentlig, i hvertfall på det.			3			
	[30:28] For når man skal ha en 8-kant, så er, ehm, for hver gang liksom så vrir han seg fra den veien og henn der, eh, 360 ganger 8. Og da må man gjøre det 8 ganger …			3		5	
Trine	[30:42] 360 ganger 8?	1					
Kari	[30:45] 360 delt på mente jeg. Så vri, så hennover, så vri og så hennover og så vri og så hennover … Og sånn fungerer det med alle mangekanter, egentlig, det har jeg aldri tenkt på.	1		3	4	5	6

8.7 Transkripsjon: oppgavebasert intervju om lineære funksjoner

		1 Utf. og pbl.	2 Mod. og anv.	3 Res. og arg.	4 Rep. og kom.	5 Abs. og gen.	6 Mat. ksp.
TC	[00:01] Ja, da er jeg litt heldig som har med meg noen som har brukt Scratch før. Og så er det viktig at dere snakker underveis sånn at jeg får høre hva dere sier og hva dere tenker. Og så er som sagt målet at dere skal lære noe om variabler.						
Erik	[00:13] Okay... Oppgave a, lag et program som spør brukeren hvor gammel hun er og som deretter svarer hvor gammel brukeren vil være om 10 år. Da må vi begynne med den derre...	1					
David	[00:26] Begynne med «start...»	1					
	(Finner blokken «Når grønt flagg klikkes».)						
Erik	[00:31] klikkes, også vi må jo ha	1					
David	[00:35] og den, også spør	1					
Erik	[00:36] skal vi se	1					
David	[00:37] Hvor gammel er du	1					
	(David skriver spørsmålet)						
Erik	[00:44] Tar du stor «h» også? Det er himla irriterende å se på når det er små bokstaver.			3			
David	[00:51] Sånn			3			
Erik	[01:00] Hvor gammel er du og vent… og da blir det jo … Vi må jo lage en variabel… som heter… (leter fram knappen «lag en variabel»). Nytt variabel navn, det er jo «født», når du er født, da skriver du bare «født».	1		3	4	5	6
David	[01:23] ja (skriver)	1		3			
Erik	[01:30] Okay, men da… er det jo… først må vi få nesten sette verdien…på de to, så da må jo den settes til 0. («sett født til 0»). Vi må også ha en til da… som het… vi trenger enda en variabel. Det er jo årstallet nå, bare skriv nå. … Så tar vi sett… Det har ikke noe å si rekkefølgen.	1		3	4		
David	[01:59] Og så…født	1					
Erik	[02:01] … og vent. Og så… da, hvis du svarer jo da… og da må vi ha utseende… nei… det var noe rart. … Variabler. Da må vi sette …… «nå», for han spør først hvor gammel er du, men da må vi forandre på det, som blir, ja, sett «født» til ... er det «sansing»? svaret der. (setter født til «svar».) Sånn og så må den spørre en gang til, men da må den spørre hvor gammel… nei, hvilket år det er nå. Tar du og skriver?	1					
David	[02:41] Mhm.	1					
Erik	[02:49] Tar du stor «h» også?	1					
David	[02:51] Eh, chill.	1					
Erik	[02:56] Sånn, hvilket år er det, spør om det. Det blir jo et svar, så må vi på variabler igjen. Så setter vi «nå» til det året. … Eh, der ja. … Okay, og deretter svarer hvor gammel brukeren vil være om 10 år. … Da har vi fått de to svarene. … Så, vi tester hvordan det er så langt, start. «Hvor gammel er du?» så skriver du	1		3	4		
David	[03:35] 15	1					
Erik	[03:36] «Hvilket år er det?» 2019. … Okay, alt fungerer så langt. … Men så… er det på… sett sammen… men først må vi ha den der (finner blokken «si») … Da må du bare ta «om ti år er du».	1					
David	[04:26] Vent litt, stor «m», nei…	1					
Erik	[04:29] Stor «o». Også, om ti år er du «banan», nei det er du ikke. Men, skal vi se… Om ti år er du, da må du ta året nå minus noe… om ti år er du, skal vi se … Det må bli 2019 minus 2004, for da blir det et positivt tall i stedet for negativt.	1		3	4		6
David	[04:54] Mhmm.	1					
Erik	[04:54] Og da blir svaret riktig. … «Nå»	1					
	(Finner «nå» variabelen, men den plasseres ikke som tenkt.)						
David	[05:06] Ey!	1					
Erik	[05:06] Der skjedde [???] (retter opp i feilen)… minus «født». Den må ut, for vi må jo legge til… pluss… skriv ti. … Jeg håper det funker, for det er det jeg lærte på programmering.	1					
David	[05:33] Da prøver vi… 15	1					

Erik	[05:37] Jeg håper ikke vi har glemt noe. … alt skal være riktig. … Jeg tror du bare kan trykke «enter»	1					
David	[05:49] Det funket (å trykke enter).	1					
Erik	[05:50] Om ti år er du 1994	1					
	(Latter!)						
David	[05:53] Nei, nei , nei…	1					
Erik	[05:54] Åja, det er jo fordi du har… må vi plusse på 10 først? Så da tar du… «Om ti år» kan du legge til mellomrom (mellom tekst og tall). … [???] fordi, den vil jo først plusse sammen, så vi tar minus etterpå. … men da må vi… skal vi ha det i en egen snakkeboble? Eller må vi til med parentes her? … nei	1		3	4		
David	[06:36] Shii, jeg vet ikke. … Kan vi ikke bare bytte på de?	1		3			
Erik	[06:44] Tror ikke det går, for du tar født + 10 år … JO!	1					
David	[06:47] Hva er det det står her? (ser på arket) Vi må bare bytte…	1					
Erik	[06:49] Det funker.	1					
David	[06:50] Her, her står det ikke noe pluss, though. Eller nei, ja, vi må jo plusse	1					
Erik	[06:54] Det er eksempelet, de gir oss jo ikke svar på oppgaven. … født pluss 10 minus … okay, hvis vi tar dette … det blir et negativt tall, så disse to må bytte plass. Hvis vi tar «nå» pluss 10, ja det skal jo stemme.	1		3			
David	[07:14] Skriver 10	1					
Erik	[07:16] For du tar jo nå, 2019, 2029 blir det jo da, minus 2004 som vil være noe… Det er ikke … eh…	1		3			
David	[07:27] Men bare prøv.	1					
Erik	[07:30] Hvor gammel er du, skriv 15. Oi, begge trykket samtidig.	1					
David	[07:37] Åja…						
Erik	[07:40] Hvilket år er det? 2019.	1					
David	[07:45] What the … Vent litt. …						
	Erik legger hodet ned på bordet						
Erik	[07:57] Jeg innså hva vi gjorde feil.	1					
David	[07:58] Å ja.	1					
Erik	[08:00] På «født» skrev vi 15 år.	1		3			
David	[08:03] Å ja, shii… Skal vi forandre på den da?	1		3			
Erik	[08:08] Ja, nei, du må… Vi må bare skrive «hvor gammel er du?» 2004	1		3			
David	[08:13] Å ja (ler)	1		3			
Erik	[08:14] Hvor gammel er du?, vi greide å forvirre oss selv. … Det er jo genialt av oss.	1					
	(svarer 2004 på «Hvor gammel er du?»)						
Erik	[08:21] Så det funket!	1					
	(David klapper)						
Erik	[08:24] Det var bare vi som er …	1					
David	[08:25] Skal vi forandre på verdi… variabelen	1					6
Erik	[08:29] Vi tar heller bare «Hvilket år er du født?» isteden.	1			4		
David	[08:32] Hehe, så vi bare avoider problemet. …. Eh, stor «h».	1					
Erik	[09:10] Vi tar noe annet, hvis vi skriver, la oss si at…	1					
David	[09:13] Vi sier vi er 03.	1					
Erik	[09:19] Også skriv bare 2052.	1					
David	[09:22] Eh, ok…	1					
Erik	[09:23] Jeg tok bare tilfeldig tall. … Om ti år er du 59 år, det skal stemme ganske riktig det.	1					
David	[09:32] Ja.	1					
Erik	[09:36] Ja, men det stemmer faktisk nøyaktig. Da funker det. … Så oppgave b) da.	1					
David	[09:41] Oi, det er oppgave b).	1					

Erik	[09:44] Okay, lag et program som spør brukeren om fornavnet sitt, deretter om etternavnet sitt og som deretter svarer det fulle navnet til brukeren. … Skal vi se, da kan vi egentlig… Hvis vi bare, sånn, så kan vi spare koden (løsner koden fra «når grønt flagg klikkes»)	1					
David	[10:07] Ja	1					
Erik	[10:08] Det er ikke vits i å kaste bort god kode.	1					
David	[10:13] ... og så spør.	1					
Erik	[10:16] Ja, og nå blir det, «hva er fornavnet ditt?» … Stor «h».	1					
	(David skriver)						
Erik	[10:35] Sånn og så, nye variabler. Den her kaller vi bare fornavn. Litt irriterende når jeg ikke har tastaturet her foran… men ok… da er det jo enda mer samarbeidsoppgave. … Vi kan jo egentlig skjule «nå» og «født» variablene, de bruker vi ikke akkurat nå.	1		3	4		6
David	[11:01] Vi kan jo egentlig skjule alle.	1					
Erik	[11:11] … Okay, da har vi fornavn. Men da må vi jo… variabel. Sett fornavn til, så må vi finne den svar klossen, som er, der. Fornavn til svar. Så må det, da må den etterpå spør, etter å ha settet fornavn til … Da må vi lage enda en variabel, men først kan vi sette fornavn til 0, og så… lag en variabel som heter etternavn. Skal vi legge inn mellomnavn og eller?	1		3	4	5	6
David	[11:40] Nei, eller. Hvis vi har sånn ekstra…	1					
Erik	[11:44] tid?	1					
David	[11:44] mhm.	1					
Erik	[11:47] Spør, hva er etternavnet ditt, må det jo være.	1					
	(David skriver)						
Erik	[12:09] Fint, da har vi det. Sett etternavn til 0. Koble sammen de to. Spør først, gi svaret, hva er etternavnet ditt? Da må vi ta å sette etternavnet til det som vi får som svar. (Gestikulerer med fingeren, først dette, så dette…) Så må han si… Nå kan vi i hvertfall ikke rote til med de tallene. Skal vi se, «sett sammen». Da må vi sette sammen…	1		3			
David	[12:40] på variabler	1			4		6
Erik	[12:41] Sett sammen fornavn … Nå tester vi bare, hvis du bare skriver	1					
David	[13:00] Oi, mellomrom, shiii. … Eh, ehm. Må vi lage, nei vi må ikke…	1					
Erik	[13:07] Hvis du bare tar mellomrom… vent litt… det må da…	1					
David	[13:11] Men kanskje hvis du bare kaller det for, for, nei… bare glem det.	1					
Erik	[13:16] Nei, hvis du bare skriver… det funker da, men hvis vi bare, du bare skriver mellomrom…	1					
David	[13:20] Å ja, jajaja. (skriver fornavnet med mellomrom på slutten)	1					
Erik	[13:30] Sånn, nå ble det riktig, men vi kunne også egentlig bare lagt til en variabel som bare hadde vært et mellomrom, så hadde den kommet automatisk. Tror jeg hadde vært bedre. Men den funker den nå da, så da er det jo, skal vi se… Følg linken…	1		3	4	5	6
TC	[13:45] Den er klar på neste fane.						
Kor	[13:45] Ok.						
David	[13:47] Oi… oi.						
Erik	[13:48] … som ble vist først i timen, her er det seks tomme bokser… okay. «pek i retning 90, sett input til 0, output til 1» … jo, det er det vi ser her. Se her vet du.	1					
	(De teller de tomme boksene)						
Erik	[14:17] Okay, men hvordan koden funker… Den vil jo først, den vil snu seg den veien. Så outputtet blir [??]. Å ja, den der vil legge seg bakerst. Og laptoppen også har egen kode.	1					
David	[14:31] Åååå…	1					
Erik	[14:35] Vent litt, nei bakgrunnen hadde ikke egen kode den også.	1					
David	[14:37] Er det noe som er feil på laptop?	1					
Erik	[14:41] Egentlig ikke.	1					
David	[14:41] Nei.	1					

Erik	[14:44] Men det er hvertfall… så er det jo… ja, han vil gå bakerst og bli mindre, så vil den skjule, nei, alt det der… og så vil den gå der, bortover sånn, bli vist, spør «velg et tall», er det startkoden, alt det der, og da vil den ende opp der. Sånn at den blir lagt bak laptoppen og alt det der. Det trenger vi ikke fokuser på, men. Spør «velg et tall og vent». Da setter vi input til det svaret vi får.	1		3			
David	[15:15] mhm.			3			
Erik	[15:19] Og så, si … Hva skal den si? … okay, han spør om han kan velge et tall, og så setter, blir input svaret, så vil han … Jo, øh, det stod jo i timen, det der beregner eller noe sånn. … Vi kan jo bare skrive rett inn der, hvis du bare skriver beregner… sånn.	1					
TC	[15:59] Trykk på det grønne flagget, så ser dere hvordan det ser ut foreløpig. Så kan dere prøve å huske hvordan det så ut inne i klasserommet.						
Erik	[16:03] Velg et tall	1					
David	[16:04] mhm, åja… Vi sier… 3.	1					
	(katten sier «beregner» passerer laptoppen som sier «behandler»)						
Erik	[16:12] Nei, behandler.	1					
David	[16:13] Å ja, hehe.	1					
TC	[16:33] Inne i klasserommet, så sa katten det tallet som dere valgte.						
David	[16:39] Å ja. Si sv… nei	1					
Erik	[16:44] Jo, si svar.	1					
David	[16:45] Ja, svar.	1					
Erik	[16:49] Åja, det er sant det for den gikk fra over der og kom ut med svar. Svar. Så kommer den ned der, ikke sant, output vil være svaret, eller vi kunne også brukt input der, … gange to pluss en, så … sånn…	1		3			
David	[17:23] Og så si …	1					
Erik	[17:25] Da vil den si…	1					
David	[17:26] Output.	1					
Erik	[17:31] Så vil den… jo, jo, vent litt. Gå til kanten, sett sammen, sett sammen. Da blir det svaret…	1					
David	[17:42] Nei, nei, nei… Det vi sa først.	1					
Erik	[17:45] Ja, det er svaret, svaret vil forbli der, fordi output…	1					
David	[17:48] Ja, svaret og så output, ja ja ja.	1					
Erik	[17:56] Skal vi teste den? … Velg et tall.	1					
	(David velger 3)						
David	[18:05] Ja, det stemmer. … Eller, skal ikke den være en opp, slik at det blir en pil (peker på «pilen» mellom verdiene)	1					
Erik	[18:12] Jo, men… det er nok litt forskjellig fra PC til PC. … Men da var den ferdig.	1					
David	[18:18] Var det en d)?	1					
Erik	[18:19] Gjør din egen endring på «funksjonsmaskinen» slik at den regner ut andre verdier enn originalen for eksempel 2 blir til 8… Det er ingen bakside her, så da er vi, bare gjør det.	1					
David	[18:29] Oh, hva er det vi skal gjøre da?	1					
Erik	[18:30] Det hadde vært mye gøyere å holde på med dette her i matten.						
David	[18:32] Ja, gange… Hvor mye skal vi ta gange?	1			4		
Erik	[18:34] Her kan vi bare ta…	1					
David	[18:35] Ja, bare forandre på den.	1					
Erik	[18:36] Vi kan også ta deling da? … og forandre på denne? … Men hvis vi tar, la oss si, gange med …	1			4		
David	[18:46] 5	1					
Erik	[18:46] 3 pluss 5 eller et eller annet sånn.	1					
David	[18:48] Ja	1					
Erik	[18:48] Bare skriv tilfeldige tall… Ikke så høyt men… 5 pluss … sånn … så må vi teste, det skal jo egentlig funke men.	1					
David	[19:02] Så for eksempel 10	1					

Erik	[19:04] Da skal svaret bli 53.	1					
David	[19:06] Gange, ja, 53. (Det stemte)	1					
Erik	[19:08] Da funker det. Da er egentlig alle oppgaver gjort… ja.	1					
TC	[19:14] Hva endret dere på nå? Det siste?						
Erik	[19:16] Vi endret på at i stedet for at det står ganger med to, så tar det svaret og ganger det med 5 og så legger til 3 i stedet for å legge til 1.	1		3	4	5	
TC	[19:26] og da testet dere med?						
Erik	[19:28] 10, og da er det veldig lett å regne ut om det stemte eller ikke.	1					
TC	[19:32] Og 10 ble til?						
Erik	[19:33] 53	1					
TC	[19:34] Kan dere lage et nytt eksempel?						
David	[19:41] Bare trykk på det grønne flagget… 20… 103	1					
Erik	[19:51] Ja, det stemmer. Ta et tall som er litt vanskeligere å regne ut i hodet, se om det funker fortsatt? 40000605.	1					
David	[20:04] Ehm, ja … noe sånt. (6947 -> 34738). Ja, vi antar at det stemmer.	1					
TC	[20:14] Kan dere lage et tredje eksempel på en slik funksjonsmaskin?						
Erik	[20:18] Eh, skal vi gå inn og endre?	1					
David	[20:31] Skal vi legge til noe minus? Sånn at det blir gange 5 pluss 3 og så minus… 100 eller noe sånt?	1					
Erik	[20:37] Vi kan også bare ta, i stedet for	1					
David	[20:39] Ja, ta delt på. I stedet for gange.	1					
Erik	[20:43] Da risikerer vi at det blir kommatall, men det får gå. Jeg legger bare den der midlertidig (flytter multiplikasjonsklossen), operatør, det er deling… Svaret delt på…	1		3			
David	[21:01] To, skal vi ta to? Da blir det enkelt.	1					
Erik	[21:05] Så… For å teste og [???] så skriver vi bare 10. Som svar. For det er greit å regne ut.	1					
David	[21:01] Det burde da bli…	1					
	(Katten viser 8)						
Erik	[21:19] Ja, det stemmer… Da vet vi at alt det her funker.	1					
TC	[21:28] Kan dere prøve et nytt tall.						
David	[21:31] Liksom oppe? Eller i koden?	1					
TC	[21:33] Der ja.						
	(100 -> 53)						
Erik	[21:41] Det stemmer.	1					
TC	[21:50] Okay, da vil jeg … Er dere kjent med GeoGebra?						
Erik	[21:55] Vi har hatt det i matten før ja.						
TC	[21:57] Ja, da kan dere… Gå inn på GeoGebra. (Åpner programmet.) Er dere kjent med å skrive inn koordinater?						
Kor	[22:29] Ja						
TC	[22:31] Da kan dere begynne med å skrive inn der oppe, parentes… så ser vi på (gir dem et ark med tallparene vi har fått til nå) mitt eksempel. Hvis dere sa 2 var resultatet 5, så tolker vi dette som et koordinatpar, (2,5).						
Erik	[22:51] Ok. To, komma, fem.	1					
David	[22:59] Skal vi se, tre komma syv.	1					
TC	[23:07] Okay, og så hadde vi noen flere, men de tallene var så høye. Eh… Husker dere hva som skjedde her? Funksjonsmaskinen?						
Erik	[23:20] Svaret, ganget med 2 og la på 1.	1					
TC	[23:22] Så hvis vi hadde valgt fire her da? … Hva blir resultatet?						
Erik	[23:25] 9.	1					
TC	[23:27] Og hvis vi velger 5?						

Erik	[23:30] fem, ti, elleve!	1					
TC	[23:32] Kan dere skrive de også inn som koordinater?						
David	[23:36] fire komma ni.	1					
Erik	[23:43] Jeg tror jeg vet hva som kommer til å skje	1					
David	[23:46] Ja. Fem komma elleve.	1					
TC	[23:52] Og hva er det du tror kommer til å skje, sa du?						
Erik	[23:54] At vi fortsetter med samme mønster oppover. Istedenfor å bruke den maskinen der, kunne du bare sett at du skal en bort og to opp, og da ville … svaret… du ville hatt 6 og fått ut 13 (peker på koordinatsystemet). Så da vil du kunne brukt GeoGebra eller koordinatsystemet til å finne ut hva svaret ville vært før du hadde tastet inn i maskinen. … Det er vel et verktøy, hvis du for eksempel hadde tatt den linja så ville den gått oppover der…	1	2	3	4	5	6
TC	[24:23] Ja, bare tegn inn den linja hvis du vil.						
	(Tegner linje mellom to av punktene)						
Erik	[24:44] Si at vi for eksempel tok 12, da ville det blitt 25. (Bruker linja til å forutse verdier) Jeg kan se svaret her ut i fra den linja.	1		3			
TC	[24:52] Okay, hva hvis vi ser på den første maskinen som dere lagde? Da ble 10 til 53 bl.a.						
David	[25:10] ti komma femtitre…	1					
Erik	[25:14] da vil vi vel også kunne forutse hvilke tall maskinen vil regne ut.	1					
David	[25:28] tjue komma hundreogtre	1					
Erik	[25:21] Hvorfor tok vi egentlig så høye tall? (ler) … tjue komma hundreogtre, var det ikke det?	1					
David	[25:25] Ja.	1					
	(Blar og zoomer i grafikkfeltet for å finne disse punktene)						
Erik	[25:57] Hvis vi hadde tatt en linje mellom de to punktene så vil vi kunne valgt tall og der de… ja… vi kan se på et eksempel her.	1			4		
	(zoomer inn slik at punktet E er tydelig markert på rutenettet).						
Erik	[26:12] Det krysningspunktet der. Så ville vi f.eks. kunne tatt 11 in og funnet ut at det blir 54 uten å trenge å bruke maskinen.	1		3	4		
TC	[26:21] La oss prøve.						
Erik	[26:22] Eller det kan være at linja blir litt annerledes… Jeg kan trekke en linje så blir det litt lettere å se, for det blir sannsynligvis litt feil. … Eller mest sannsynlig bli et annet tall.	1		3			
David	[26:40] Det burde bli 58.	1					
	(Blar og zoomer i grafikkfeltet til de kan lese av at det stemmer med (11, 58).						
TC	[27:20] Hva tror dere om den neste da? (Peker på tredje funksjonsmaskinen)						
Erik	[27:25] Da må vi for det første skrive inn punktene så kunne vi også finne ut hva den ville svart.	1					
TC	[27:31] Hva tror vi på forhånd kommer til å skje?						
David	[27:34] At det blir samme linje. Rett linje.	1			4		
Erik	[27:38] Det går en linje mellom punktene, og igjen vil de krysse der, og da kan vi si… hvis det for eksempel hadde vært at 12 hadde blitt til 54 så kunne vi finne det ut uten å måtte taste det inn i maskinen eller regne det ut. Fra den dataen vi allerede har kunne trekke en linje og funnet ut hvilke tall vi ville fått. Vi kunne forutsi hva maskinen ville svart.	1		3	4		
	(Taster inn koordinatene (10,8) og (100,53), ordner grafikkfeltet, tegner linje mellom punktene.)						
TC	[28:21] Da må vi dobbeltsjekke, velg en verdi.						
Erik	[28:23] Kan ta, hvis jeg kan finne en. (zoomer og drar i grafikkfeltet.) Der! … 106 ville blitt til 56. … Og det kan vi teste med maskinen hvis vi skriver inn 106. (endrer vindu til Scratch, tester)… Jepp, det ble riktig.	1		3	4		
TC	[29:04] Vil det alltid bli ei rett linje?						
Erik	[29:09] Det vil ikke alltid bli en rett, men det vil alltid treffe ett av de her punktene. Det vil alltid være krysningspunkt der. Og den linja vi trekker vil treffe et kryss mellom to koordinater, sånn.	1		3			
TC	[29:25] Hva hvis vi bruker desimaltall? (Erik pekte på heltalls rutenettet)						

Erik	[29:28] Da ville vi fått sånn der (zoomer inn slik at verdiene langs aksene er desimaltall). Der står det 102,5, den ville kommet på midten der og blitt kommatall. Eller desimaltall sånn sett da. Jeg kan zoome enda mer inn så får vi se desimalene. Da ville for eksempel… Litt vanskelig å se… Hvis vi tar, ja, 154,2… eller… Hvis du hadde tatt 102,4 så ville det ha blitt 54,2.	1		3			
TC	[30:02] Dobbeltsjekk med maskinen.						
David	[30:06] Var det hundre og .. nei, hundreogto komma fire,	1			4		
Erik	[30:10] Jeg mistenker at det kanskje blir litt feil, men…	1					
	(Tester med funksjonsmaskinen i Scratch)						
David	[30:21] Tre! … Kanskje delt på…	1					
TC	[30:26] Prøv punktum i stedet for komma.						
	(Tester igjen)						
David	[30:40] Ja! Riktig.	1					
TC	[30:44] Så hvis når de andre elevene i klassen har gjort samme greia, de har også endret… Vil de også få rette linjer i GeoGebra?						
David	[30:51] Jaaaa…?	1					
Erik	[30:53] Hvis de har gjort nøyaktig samme så vil de også få rette linjer og hvis de hadde tatt… brukt for eksempel det eksemplet der, eller ditt, et av de der, og vi hadde satt den linja, så ville vi kunne uten å ha brukt maskinen funnet ut hvilket tall det ville blitt. Og det ville blitt riktig hvis vi.. den hadde variert ut ifra hvilket eksempel vi brukte hva svaret hadde blitt.	1		3	4		
TC	[31:18] Da lurer jeg på om dere kan lage funksjonsmaskinen på en sånn måte at resultatet, altså koordinatene, ikke blir ei rett linje? Er det mulig eller er det umulig?						
David	[31:30] Hmm…	1					
Erik	[31:33] Det vil være umulig for på et eller annet tidspunkt så vil de linjene treffe hverandre for de er ikke…. Ja, hva heter det igjen?... De er ikke parallelle, de blir bare på rett sånn (viser med hendene) og da vil du på et av krysningspunktene så vil du på et eller annet vis treffe…	1	2	3	4		6
David	[31:48] Vi kunne jo se at her er det sånn 2 tall (peker på dividert på 2 i koden) også kunne det ha vært sånn tilfeldig til hvilken det byttet…	1		3			
Erik	[31:55] Det kunne variert på den…	1		3			
David	[31:59] Men da er det tilfeldig, da er det ikke noe sånn rytme, eller.	1		3	4	5	6
Erik	[32:02] Da ville det ikke vært mulig… Men du ville jo hatt det på noen. Kanskje til og med da på noen punkter hatt sånn der [???] men det ville ikke vært helt sikkert. … Vært masse tilfeller hvor det [???] … Det er jo faktisk et verktøy i GeoGebra da (leter fram «beste tilpasset linje») som lager ei ca. linje mellom mange punkt. Det er sånn man bruker for å si at ja, det vil være noe innenfor der. Det der eksempelet hvis du hadde tatt… ja forskjellige møn… Men vi brukte jo et eksempel for å skulle forklare det etter machine learning, ved å tatt forskjellige, sånn der [???] så ville du ut ifra hvor store de var og hvor komplekse de var så kunne du finne ut hva som er likt og ikke likt, og tegnet ca. grenser. … Men hvis vi bare bruker, ja, helt vanlige tall, så ikke bare [???], da ville det være umulig å kunne trekke en linje uten at vi vet. Hvis vi hadde brukt variabler der i stedet, og da tatt et tilfeldig tall, da ville det vært umulig, nei, ja da ville det vært mulig. …	1	2	3	4	5	6
TC	[33:18] Da har jeg to alternativ til dere. Nå snakker vi om at det er mulig, hvis vi hadde lagt til noen andre variabler?						
Kor	[33:23] Mhm.			3			
TC	[33:24] Da kan dere enten forsøke dere på det, eller så snakket du også om parallelle linjer, og hvis vi da går i GeoGebra, og ser på de tre linjene (zoomer ut og drar i graffikkfeltet). Okay, de er ikke parallelle. Kan du lage en ny sånn funksjonsmaskin som lager ei parallell linje til en av disse?						
Erik	[34:00] Det kunne kanskje vært lurt, vi kan se på, eh, ny, det var den der… Da måtte vi i så fall fått 10 til å bli… Den måtte gått sånn da	1					
David	[34:18] Ja	1					
Erik	[34:18] For eksempel 12… Så hvis vi går utgangspunkt i at vi må få 10 til å bli 12. Og 12 blir 14. Da vil det være, ja, for eksempel, 10 ganger 1 pluss 2.	1		3			
David	[34:33] Mhm.	1		3			
Erik	[34:34] Vent, så… Ja, det vil faktisk stemme. Hvis du tar svaret, eller, ja svaret ganger 1 pluss 2, eller, du kunne også bare tatt svaret pluss 2. Så ville du fått ei parallell linje som ville gått fra, ja,	1		3	4		6
David	[34:51] Skal vi prøve?	1					
Erik	[34:52] Nei, vent litt… Du har to, den ville truffet der, fire den er nødt til å gå over…	1					

David	[34:56] Å nei.	1					
Erik	[34:59] … men ville truffet der. Det skulle teoretisk sett vært mulig. Hvis jeg tar, lager to punkter som er… skal vi se, der og… Skal vi se om det blir parallellt eller ikke. (tegner to nye punkt og trekker linje mellom).	1		3	4		
David	[35:25] Ja!	1					
Erik	[35:25] Ja, parallell. Da måtte 10 bli til 10, 14 bli til 12, har vi noe mønster ut av det? … 6 til 8… På et eller annet tidspunkt vil det bytte retning, så jeg tror ikke det er helt mulig men.	1		3		5	
David	[35:44] Vi må bare finne forholdet til de to tallene.	1					
Erik	[35:48] Problemet er at det varierer, for du ser at 2 blir til 6, 4 blir til … nei, det blir desimaltall, må zoome ut da. … 2 blir 6, 3 blir 6,5 ca. , 4 blir til 7, 6 blir til 8, 8 blir til 9, 10 blir 10, 12 blir til 11, nei vent litt… pluss fire, pluss 3, pluss to, pluss 1, pluss 0… Der blir det minus, så det ville byttet, 10 er midtpunktet. Det ville byttet fra pluss til minus, så det ville vært litt vanskelig, men det kunne kanskje vært mulig, men det tror jeg ikke vi får til akkurat nå.	1		3	4		
TC	[37:23] Prøv en annen linje da.						
David	[37:32] Skal vi begynne fra 0?	1					
Erik	[37:33] Vi kan ta den her. 1 blir til 1, 2 til 2, 3 til 3, nei her blir det bare å gå… opp til to… Hvis jeg trekker linja. Den bli til den, en bort og to opp…	1			4		
David	[37:58] Kanskje hvis vi går på Scratch, også deler de i to, tallene to, da blir det jo…	1					
	(Erik fortsetter å tegne den linja han startet på)						
Erik	[38:20] Der har vi for eksempel den. For jeg sa 1 blir til 2 der. 2 blir til 4.	1		3	4		
David	[38:26] 3 blir til …	1		3	4		
Erik	[38:28] 3 blir til 6, så da ganges det med 2, så den vil det være mulig å [???]	1					
	(Endrer koden i Scratch slik at funksjonsmaskinen bare multipliserer med to uten å legge til noe)						
David	[38:53] Hvilken var den andre? (Hvilken funksjonsmaskin tilhører linja de nå lager en parallell til?)	1					
Erik	[38:58] Vi holder på med den nå, vi tester vel den (peker på ark med resultatene fra funksjonsmaskinene). … Ta et tall nå som ikke er så høyt. (Tester sin nye funksjonsmaskin) … 10 burde bli til…	1			4		
David	[39:19] 20.	1					
Erik	[39:21] Og da har du jo… 10 er der, og så… (må dra og zoome for å få 20 langs y-aksen) 10 blir til 20, så da er det riktig.	1					
David	[39:51] Ja.	1					
TC	[39:57] Og da er dere på den (peker på ark) og funnet at den er parallell med den (peker på skjerm). Hvis vi hadde hatt mer tid, ville jeg ha utfordret dere på hvorfor de blir parallelle? Om dere kan se det på forhånd, er det noe som er felles med uttrykket der, og det uttrykket dere har lagd der?						
David	[40:11] Begge er ganger to.	1					
TC	[40:14] Begge er gange to.						
Erik	[40:17] Bare der har vi tatt min…, vi har tatt bort 1, så da blir det flyttet et hakk ned. Egentlig så ville linjen gått sånn, men siden vi flyttet den et hakk ned vil den forskyve seg bortover her.	1		3	4		
TC	[40:30] Så du mener at du kan lage enda en parallell linje?						
David	[40:32] Ja.	1					
Erik	[40:32] Mhm. Da kan vi ta, i stedet for, pluss 1 da tar vi pluss 2.	1		3	4	5	6
David	[40:37] Mhm.			3			
	(Bytter til Scratch, endrer uttrykket)						
Erik	[40:44] Så, hvis vi bare tar. Bare skriv inn to tall. (Tester maskinen.) 10, det vil jo bli 22 (snakker samtidig som katten viser resultatet.) Også, ta enda et tall. … 12, 26… Skal vi legge inn som koordinater (bytter til GeoGebra, skriver inn punktene, tegner linje.) Så hvis jeg trekker linje fra M til N så vil den også være parallell.	1			4		
TC	[42:42] Javisst!						
Erik	[42:43] Så vi beholder den der, ja, ganging og deling, men vi forandrer på… Ganging eller deling, samme tall blir valgt, men vi forandrer på pluss og minus, Og når det er pluss, ja, det går oppover (linja forskyves oppover), eller om det går nedover, eller, tallet som blir plusset på. Så vi kunne, der ser du at det er minus, vi kunne for eksempel tatt pluss 5, da ville den ha flyttet seg 5 bortover sånn.	1		3	4	5	
TC	[43:15] Eh… Hva tenker dere om koblingen mellom matematikk og programmering?						

Erik	[43:21] Det er gøyere med programmering og matte, enn bare matte.						
David	[43:27] Ja.						
Erik	[43:27] For i programmering da kan vi jo ha sånn der, lage sånn der maskin og så kan du prøve å finne ut av om det er noen sammenheng, alle de forskjellige svarene, og om du klarer å lage maskin som bare flytter litt på forskjellige svarene som blir gitt. … I stedet for å bare sitte og holde på med masse teori, så kan vi jo sitte og, jobbe med sånn der litt mer praktiske oppgaver og du får for eksempel den her oppgaven hvor du skal lage en maskin, så skal du finne ut hva er sammenhengen mellom det her og om du kan regne det ut på en eller annen måte uten å bruke maskinene, om du kunne se et mønster.			3			
TC	[44:10] David, noen tanker?						
David	[44:11] Eh, ja, at, vi bruker veldig mye sånn teknologi i hverdagen som har veldig mye med programmering å gjøre, så hvis man lærer litt av det på skolen, så er det, kan det være veldig nyttig. Fordi flere jobber i fremtiden sier jo at veldig mye om programmering.						
TC	[44:33] Hva med matematikk faget nå da? Er det ikke nyttig?						
David	[44:38] Jo, men det blir mer og mer programmering liksom, så, det er veldig få folk som kan sånn programmeringsspråk.						
TC	[44:51] Ja, da takker jeg for hjelpen. Godt jobbet!						